Image Captioning

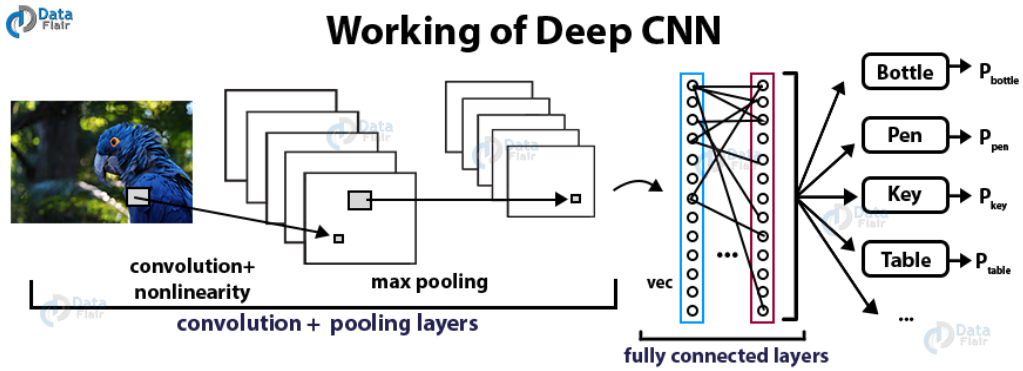
Submitted By : Preetam Keshari Nahak
116CS0205

1. Introduction

Image captioning is an application of both computer vision and natural language processing techniques which looks at generating automated short description/caption for a given image. This techniques can be used in many applications such as suggestions in a editing app, social media, for visually challenged persons.

2. Theory

This is a supervised learning techniques which incorporates combined usage of convolutional neural networks and recurrent neural networks to generate captions. A generic architecture for image captioning looks like :

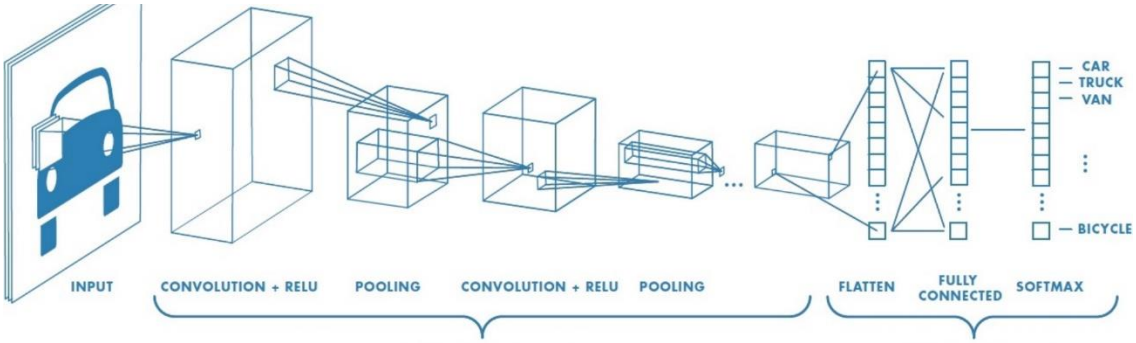


[Fig 1. Basic image captioning architecture src : Dataflair]

Basically, CNN is used to extract features of the images in vector form of fixed size. Similarly word embedding techniques are used to prepare vector representation of words in a caption. Then the image feature vector combined with caption vector data are fed into a model and trained to predict the next sequence of word one by one. Finally the trained model is used to predict accurate captions for a given image.

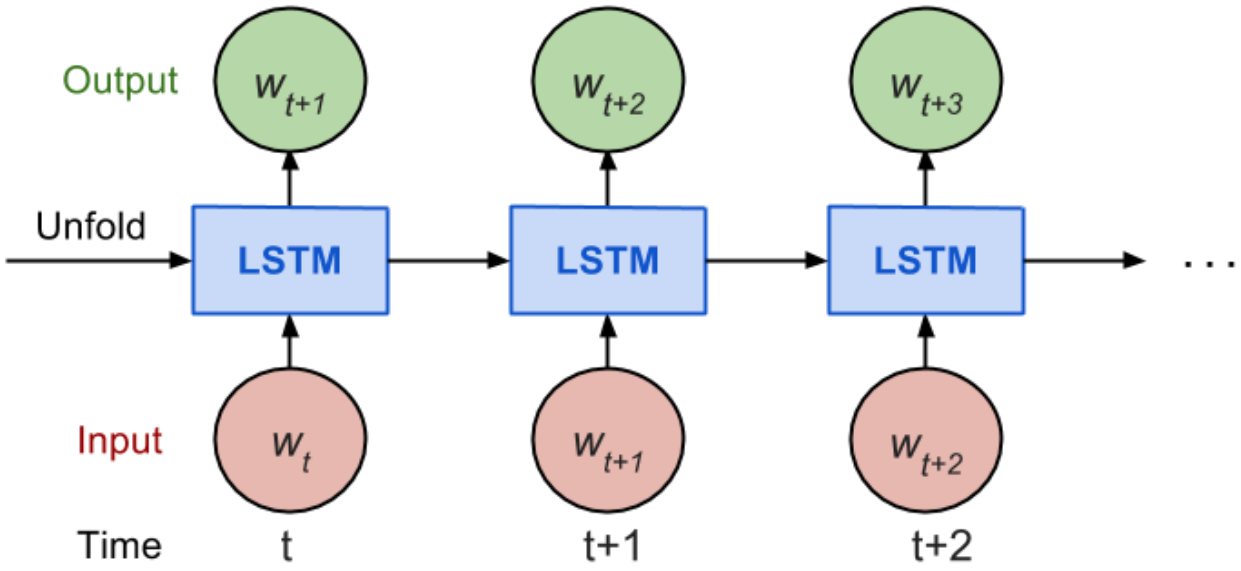
Basically a CNN consists of a convolutional layer which will use feature detectors to generate feature maps, then a pooling layer; this can be either

a max pooling or min pooling or average pooling layer, a fully connected dense layer and a flattening layer to convert the feature into a vector.



[Fig 2. Layer involved in a CNN]

Similarly, a RNN includes a embedding layer which performs the word embedding task, LSTM cells layer in order to extract sequential relation of the features and an output layer with either softmax or sigmoid activation function incorporated with it.



[Fig 3. RNN with LSTM cells]

Main challenge lies in data preprocessing phase in case of image and textual data and data generation phase. Below we discuss about these in a detailed manner

3. Methodology

3.1 Data preprocessing of image

Images will act as input to our model and hence we need to process these images to generate respective feature vectors of fixed size. For this purpose we will use InceptionV3 model by google to generate feature vector of images of size 2048. The InceptionV3 is trained on a image dataset with 1000 different classes of images. So we remove the last softmax layer and extract the desired 2048 length vector for each image.

3.2 Data preprocessing of captions

First, we prepare a vocabulary set of most frequently used words out the words included in the captions. Then we determine the maximum length of the caption so that our input and output caption length is pre- determined and fixed. Also we will maintain a word-to-integer and integer-to-word mapping of the words of the vocabulary.

3.3 Data preparation

Here we demonstrate an example of how our training data is prepared. Say we have 3 images and 3 captions with us. We keep 2 image and 2 caption for our training set and rest one for testing.

Say these training images are img_1 and img_2, captions are cap_1 and cap_2 espectively. We will pad each caption with a <START> and <END> token.

Say cap_1 := “the biker with red helmet is good”

cap_2 := “bee is on the flower”

So we will pad the captions as

cap_1 := <START> The biker with red helmet is good <END>

cap_2 := <START> Bee is on the flower <END>

Hence our vocabulary will be {<START>, the, biker, with, red, helmet, is, good, bee, on, flower, <END>} and we will label the words with index such as 1, 2, 3 respectively.

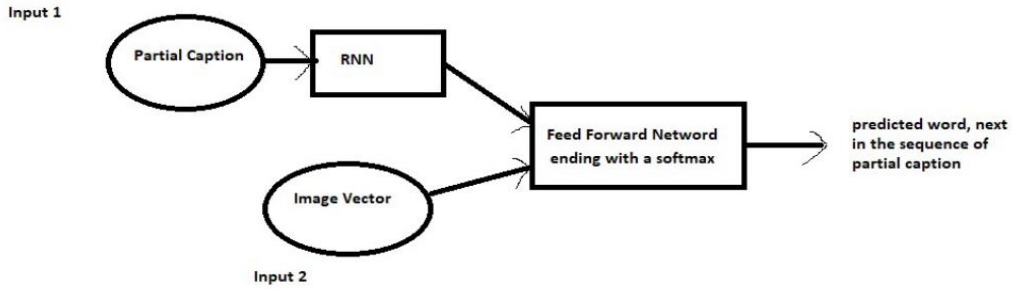
Input data will be prepared with partial captions as image data + partial caption and target data will be the next word. E.g for img_1, our data set will look like :

Xi	Yi
Img_1 + <START>	the
Img_1 + <START> the	biker
Img_1 + <START> the biker	with
Imag_1 + <START> the biker with	red
and so on…	

We will then represent each partial caption with their index form and in order to feed the network an input data of fixed length, we will pad the data with zeros. Number of zeros to be padded will be determined by maximum length of caption – number of zeros currently present. So our data set may look like

Xi	Yi
Img_1 + [1 0 0 … 0 0]	the
Img_1 + [1 2 0 … 0 0]	biker
Img_1 + [1 2 3 .. 0 0]	with
and so on…	

The img and textual data will be fed to the network in vector formats generated by InceptionV3 model and GLOVE word embedding model respectively. A high level architecture of our approach can be demonstrated as :



[Fig 4. High level view of the architecture]

3.4 Dataset

We use dataset from Flickr 8k dataset which contains about 8000 images and 5 captions each for each image. We keep 6000 image as training set, 1000 for validation and 1000 for testing data.

```
1000268201_693b08cb0e.jpg#0     A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg#1     A girl going into a wooden building .
1000268201_693b08cb0e.jpg#2     A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg#3     A little girl climbing the s
```

[Fig 5. Few sample data]

3.5 Building the model

```
In [28]: model.summary()
```

```
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_4 (InputLayer)         (None, 34)                0
_________________________________________________________________________________________________
input_3 (InputLayer)         (None, 2048)              0
_________________________________________________________________________________________________
embedding_2 (Embedding)      (None, 34, 200)           330400      input_4[0][0]
_________________________________________________________________________________________________
dropout_3 (Dropout)          (None, 2048)              0           input_3[0][0]
_________________________________________________________________________________________________
dropout_4 (Dropout)          (None, 34, 200)           0           embedding_2[0][0]
_________________________________________________________________________________________________
dense_2 (Dense)              (None, 256)               524544      dropout_3[0][0]
_________________________________________________________________________________________________
lstm_2 (LSTM)                (None, 256)               467968      dropout_4[0][0]
_________________________________________________________________________________________________
add_2 (Add)                  (None, 256)               0           dense_2[0][0]
                                                                   lstm_2[0][0]
_________________________________________________________________________________________________
dense_3 (Dense)              (None, 256)               65792       add_2[0][0]
_________________________________________________________________________________________________
dense_4 (Dense)              (None, 1652)              424564      dense_3[0][0]
=================================================================================================
Total params: 1,813,268
Trainable params: 1,813,268
Non-trainable params: 0
_________________________________________________________________________________________________
```

[Fig 6. Model Summary]

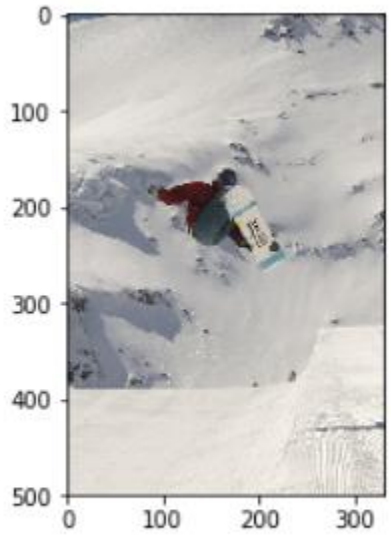
3.6 Training

```
Epoch 1/1
1000/1000 [==============================] - 68s 68ms/step - loss: 2.4380
Epoch 1/1
1000/1000 [==============================] - 68s 68ms/step - loss: 2.4300
Epoch 1/1
1000/1000 [==============================] - 68s 68ms/step - loss: 2.4200
Epoch 1/1
1000/1000 [==============================] - 68s 68ms/step - loss: 2.4085
Epoch 1/1
1000/1000 [==============================] - 67s 67ms/step - loss: 2.3994
Epoch 1/1
1000/1000 [==============================] - 67s 67ms/step - loss: 2.3957
Epoch 1/1
1000/1000 [==============================] - 67s 67ms/step - loss: 2.3877
Epoch 1/1
1000/1000 [==============================] - 67s 67ms/step - loss: 2.3766
```

[Fig 7. Training the model]

3.7 Prediction on unseen data

We use a greedy approach to predict caption for an image. i.e we start with the <START> token and image data and predict next word using our trained model. Then we feed the image data and <START> + predicted word as the input and predict next word and so on. We predict the sequence until maximum caption length is reached and return the final caption.



[Fig 8. Caption prediction]

4. Conclusion

We saw how deep learning techniques can be used along with computer vision and NLP techniques to find caption for a given image. The accuracy of the model can be increased using larger dataset, beam search instead of greedy search algorithm and obviously using a better architecture.