Visual Recognition Mini Project - Team Code AUNS

Apurva K S
MT2020011
IIIT Bangalore
sambhavi.apurva@iiitb.org

Urja Srivastava
MT2020037
IIIT Bangalore
Urja.Srivastava@iiitb.org

Nishant Chhattani
MT2020142
IIIT Bangalore
Nishant.Chhattani@iiitb.org

Soniya Jain
MT2020083
IIIT Bangalore
soniya.jain@iiitb.org

***Abstract*—Design a CNN-LSTM system in pytorch that can perform image captioning.**

***Index Terms*—CNN, LSTM, pytorch, optimizer, batch normalization, momentum.**

DATASET

Flickr8k images - A new benchmark collection for sentence-based image description and search, consisting of 8,000 images that are each paired with five different captions which provide clear descriptions of the salient entities and events. Contains a total of 8092 images in JPEG format with different shapes and sizes. Of which 7091 are used for training and 1000 for test.
Flickr8k text- Contains text files describing train set, test set. Flickr8k.token.txt contains 5 captions for each image i.e. total 40460 captions.

ARCHITECTURE

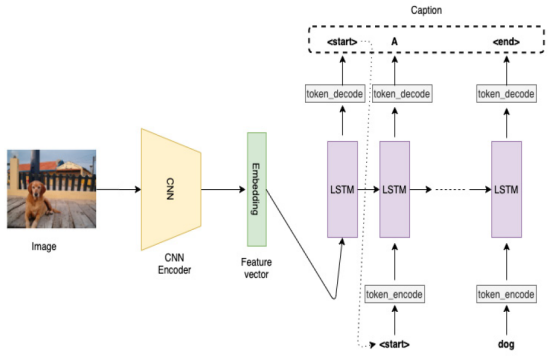

Fig. 1. CNN-LSTM Architecture for Image Captioning

We build the vocabulary which can convert word into integer tokens. The image captions are processed, one caption per image and each unique word converted into an integer token. The vocabulary object is stored for further use.
We used pretrained CNN network to output feature vectors for images. RESNET-50 was used and its last layer was removed. This last layer was trained because the output size of this layer will be the input size to the LSTM model. Thus we experimented with the size of the last layer of the Resnet-50 model. ResNet-50 is a convolutional neural network that is 50 layers deep. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. The output of this last fully connected layer is followed by batch normalization. The batch normalization layer normalizes the fully connected layer outputs with a mean of 0 and a standard deviation of 1 across the entire batch. Batch normalization helps insulate the LSTM model against any data shifts that the CNN might introduce.
The LSTM layer takes in the embedding vectors as input and outputs a sequence of words that describes the image from which the embedding was generated. The LSTM model consists of an LSTM layer followed by a fully connected linear layer. The LSTM layer is a recurrent layer, which can be imagined as LSTM cells unfolded along the time dimension, forming a temporal sequence of LSTM cells. These cells will output word prediction probabilities at each time-step and the word with the highest probability is appended to the output sentence. This is a greedy strategy to generate words. We experimented with different input layer size, hidden layer size and number of layers in lstm layer.
The final RESNET-50 encoder takes 91MB space and the final LSTM decoder model takes 8MB space. Total parameters in the RESNET-50-LSTM model were 25,909,104.

```
LSTMModel(
  (embedding_layer): Embedding(4528, 128)
  (lstm_layer): LSTM(128, 256, batch_first=True)
  (linear_layer): Linear(in_features=256, out_features=4528, bias=True)
)
```

Fig. 2. LSTM Architecture for Image Captioning

TRAINING THE CNN-LSTM MODEL

We load the vocabulary that we built and we also initialize the data loader to get train and test datasets. Using PyTorch's transform module, we perform random horizontal flip, normalization of images. We instantiate the CNN and LSTM models in the form of encoder and decoder models. We also define the loss function as cross entropy loss and the optimization with the Adam optimizer.
We run the training loop (for ten epochs) where we use the data loader to fetch a mini batch of the flickr train dataset,

run a forward pass with the mini batch through the encoder and decoder networks, and finally, tune the parameters of the CNN-LSTM model using backpropagation (backpropagation through time, for the LSTM network).
During prediction we limit the model prediction to 20 words per image input. So during prediction for any input feature vector of an image, if the caption length generated by model is less than 20 then it is padded with 0 token, otherwise we output only the first 20 words predicted.
We load the test image and first we use the encoder model to generate embeddings from the image, and then we feed this embedding to the decoder network to generate sequences. We need to convert the numeric tokens into actual text using the vocabulary object we stored earlier by applying the mapping between textual and numeric tokens in reverse. We collect these predicted captions and then calculate the bleu score between the predicted captions and original captions. BLEU is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Quality is considered to be the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" – this is the central idea behind BLEU.

Results on Subjective Images

Fig. 3. a man in a blue shirt and white pants is on a rock face .

Fig. 4. a man in a red shirt is standing next to a woman in a black coat .

Fig. 5. a man in a red shirt is standing on a rock face with a white dog .

Fig. 6. a man in a black shirt is standing in front of a large building .

Observations

The following table summarizes our results for our training experiments with 10 epochs. We also experimented with different resnet models.
Layers column- number of layers in the lstm cell
Input column- input size to lstm model
Hidden column- hidden layer size

CNN	Input	Hidden	Layers	BLEU
RESNET-152	256	512	1	0.069
RESNET-152	256	512	2	0.037
RESNET-152	256	512	2	0.029
RESNET-152	256	2048	1	0.065
RESNET-152	256	1024	1	0.065
RESNET-152	512	512	1	0.067
RESNET-50	512	512	1	0.070
RESNET-50	128	256	1	0.071
RESNET-101	512	512	1	0.064
RESNET-18	512	512	1	0.064

Test Conclusion and Bleu Score

We were able to reach a bleu score of 11%, using RESNET-50, lstm model with input size 128 and hidden size 256 and 1 layer in lstm. The model was trained for 40 epochs on training set size-7091 with mini batch size of 64 and test set size-1000 with batch size 32 was used for evaluation.

REFERENCES

REFERENCES

[1] Md Zahangir Alom1, Tarek M. Taha1, Chris Yakopcic1, Stefan Westberg1, Paheding Sidike2, Mst Shamima Nasrin1, Brian C Van Essen3, Abdul A S. Awwal3, and Vijayan K. Asari1,The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches.

[2] Michael Avendi,PyTorch Computer Vision Cookbook.