AMP Assignment 2

Visual Question Answering In Medical Domain

By: Meghana Kotagiri, IMT2014034 & G Neha, IMT2014018

1. Introduction to the Problem Statement

VQA is a multidisciplinary problem which combines two modalities: text and image. It requires computer vision and NLP techniques (probably, reasoning techniques too). The task is to answer a question correctly, where the question is accompanied by an image on which it is based.

This assignment is a domain specific challenge. ImageCLEF defines the challenge as follows: “Given a medical image accompanied with a clinically relevant question, participating systems are tasked with answering the question based on the visual image content”.

2. Dataset:

The dataset that we have used is ImageCLEF-VQA-Med Challenge dataset. The organizers of the challenge have provided three datasets: train, test and validation. The training dataset consists of 5413 medical images along with question-answer pairs. The validation dataset comprises of 500 image and question pairs. We have taken their validation dataset as our test dataset.

(Reason being that the test dataset does not have ground truths (answers) required for evaluation and we can’t find a way to submit it to the online portal.)

3. Preprocessing

a. Images:

Resize: Images are resized such that height and width of the target image is equal to 224.

Feature extraction: Pretrained VGG-Net model is used for computing the image features. The activations from the last fully connected layer of the model are extracted and used as features . The features generated are of size 4096.

b. Questions and Answers:

Tokenize: Our task here is to convert text input sequence into numerical form so that each word in the sequence is represented by a number that can be used to index vocabulary. We start by first parsing the sentence to remove punctuations, de-capitalize words etc. So, the input sentence of the form: “What does mri show?” -> “what does mri show”. Then, we tokenize this sequence to obtain: [“what”, “does”, “mri”, “show”].

Vocabulary Creation: Our next step is to create a vocabulary. We take the total number of unique words present in the questions and answers of the given dataset which turns out to be the size of 3499 in our case. We have chosen to taken all the words present in Q&A in the vocabulary instead of taking the most frequent ones. As, the answers contain medical terms, which have low frequency but are relevant in answering the question.

Additional Tokens: Also, we have added a 4 new tokens: ['<NULL>', '<START>', '<END>', '<UNK>'] that denotes “no word”, “start of the sequence”, “end of the sequence” and “unknown word” respectively to the vocabulary.

Fixed length sequences: Input sequences would be of variable lengths, so the task here is to transform all of them to fixed size. We took max question length and max answer length to be 16. The questions and answers with smaller length are padded with <NULL> token.
So the tokenized input of the form [“what”, “does”, “mri”, “show”] gets converted to [“<START>”, “what”, “does”, “mri”, “show”, “<END>”, “<NULL>”, “<NULL>”, “<NULL>”, “<NULL>”] assuming max question length is 10.

Create two dictionaries: ‘word_to_idx’ and ‘idx_to_word’. As the name suggests, ‘word_to_idx’ dictionary maps words to index of that word in the vocabulary and ‘Idx_to_word’ does the opposite.

Change to numerical form: Now, we have to transform our sequence to sequence of numbers where each number represents the index of that word in the dictionary. We can do that easily using our ‘word_to_idx’ dictionary. So our tokenized input [“<START>”, “what”, “does”, “mri”, “show”, “<END>”, “<NULL>”, “<NULL>”, “<NULL>”, “<NULL>”] changes to say, [1, 7, 28, 55, 6, 2, 0, 0, 0, 0].

4. Assumptions

We have used VGG-net model pretrained on images of other classes (mostly unrelated to the medical domain) for feature extraction, assuming that this model is capable of detecting features of medical images.

We have fixed the maximum length of answers and questions to a threshold as the most of the questions and answers have a smaller length than the longest length question/ answer in the dataset. We discarded inputs, where question/answer length greater than the maximum length.

5. Baseline Model Architecture

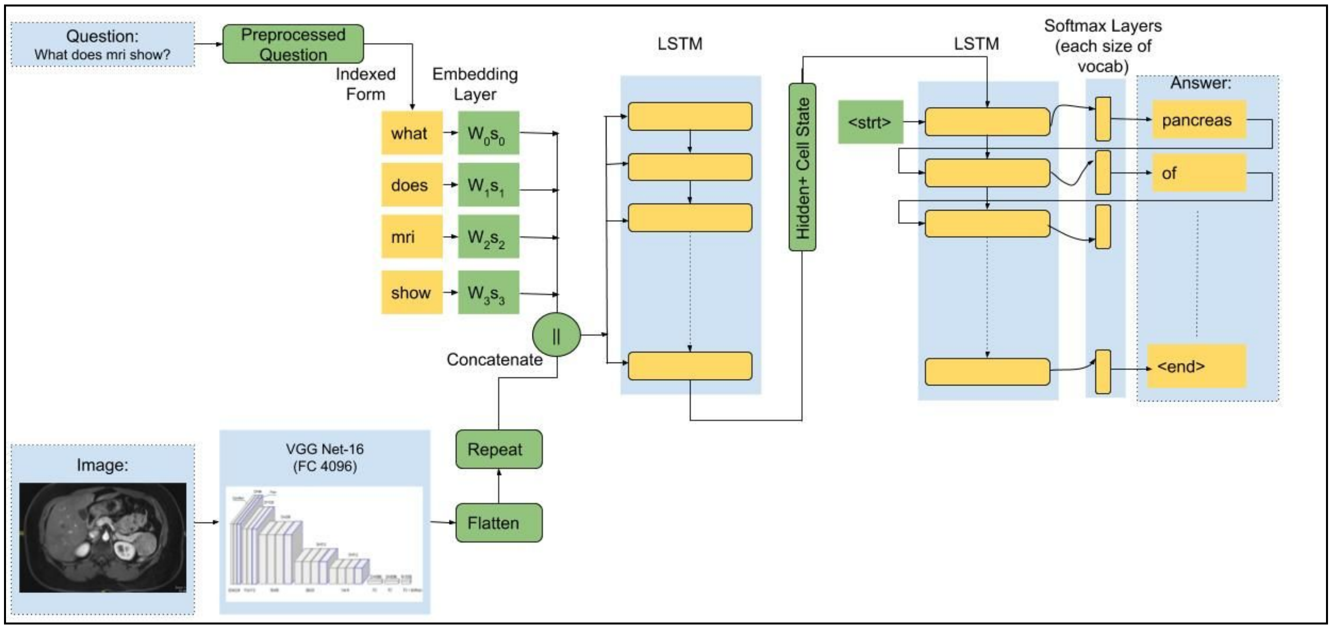


Fig.1 Baseline architecture

Our inputs to the model are preprocessed representations of our images and questions. The architecture of our model is represented in the Fig.1 above. The model has 4 modules: one for image, one for question, one to encode question & image together and one for decoding to get answer. The model has an encoder-decoder structure which we will discuss shortly.

Question Module: We use an embedding layer that learns to map input words to dimensional features (or word-embeddings). Word-embeddings help us represent our words as vectors, where semantically similar have similar word vectors. We are basically using it to appropriately encode the words. Our input to this layer that is each question has dimensions (max question length,1). The output of this layer will be (max question length, embed layer size).

Image Module: The input to the image module are features extracted using VGG-net. These features are of size (1,4096). We then use the operation repeat here to convert the features to size (max question length, 4096).

Encoder Module: The output of question module is merged(concatenated) with the output of the Image module. The idea of using repeat block for the features and merging features of the image with every word embedding of the question is to make the model learn which word corresponds to what part of the image. We then pass this merged output as input to LSTM. LSTMs are used for sequence modelling and capturing long term dependencies. We configured the model so it only outputs last hidden and cell state. Here, the (last hidden & cell state) will encode the entire question sequence information with image information.

Decoder Module: The output of the encoder module that is the (last hidden & cell state) of the LSTM is passed as the initial state to another LSTM. During training, the outputs of the decoder of the LSTM are probability distributions (over all the words in the vocabulary) generated by the model for the next word in the sentence. The model is trained to minimize the negative sum of the log probabilities of each word. During inference part, to get the next word, we pick the word with with maximum probability at each time step. The working of this part is clearly illustrated in the Fig. 1.

6. Training & Baseline Model Parameters

Drop out is applied before getting outputs from the encoder and decoder module to avoid overfitting. Cross entropy is used as loss function for training the model. Adam optimization algorithm is used for updating the parameters of the model. TanH is used as activation function in LSTM units. 20% of the training dataset is used as validation dataset.

The important model parameters and their values are used for the parameters:

- embedding size = 150
- vocabulary size = 3499+4
- dimension of hidden unit=150
- max question/answer length = 16
- learning rate = 0.001
- batch size = 100
- no. of epochs = 100
- dropout = 0.5

7. Model Extensions

For all the below variants, we have kept all the parameters same as the baseline model except the one specified

a. **Batch Norm:** Batch normalization is used to stabilize the network and it helps network converge faster. We applied batch normalization in the encoder module LSTM that is after merging the outputs of question and image module.

b. **Drop Out:** It is a regularization technique used to ensure that model does not overfit. The value of drop out parameter is changed to 0.3 in this extension.

c. **Learning rate:** Learning rate controls how model weights are adjusted with respect to the loss gradient. When learning rate is small, the model trained is more reliable but it takes time to get the optimal parameters as the parameters are updated by a smaller value. The optimizers used by us change the learning rate dynamically during the training, but we can still set the initial learning rate. In this variant, we used a smaller initial learning rate, that is, 0.0001.

d. **RMSprop:** We wanted to experiment with different optimizer to see how it effects the results. In this variant, we used RMSprop an the optimizer.

8. Results

a. Evaluation Metric

BLEU score: It captures the similarity between the answer generated by the model and the ground truth answer. BLEU scores lie in the range zero to one. Scores closer to the value one, indicate more similarity between the generated answer and the ground truth answer.

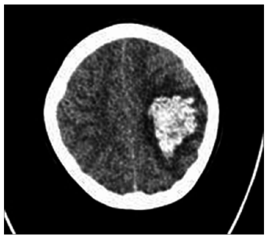
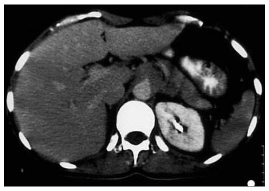
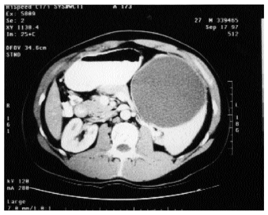
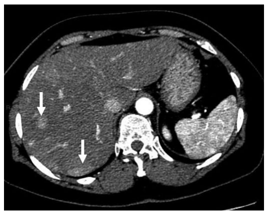
b. Quantitative results of all models

Model No	**Model variants**	**BLEU score**	**Effect on Baseline**
1	BaseLine	0.169	-
2	Batch_Norm	0.359	↑
3	Learning rate=0.0001*	0.128	↓
4	Using RMS prop as optimizer	0.269	↑
5	Drop out = 0.3	0.227	↑
	Final Model	0.393	↑

In the Final Model, we used batch Normalization, RMSprop as optimizer, dropout of 0.3 and initial learning rate of 0.001 because individually these parameters improved the BLEU score.

*Generally, decreasing learning rate should improve the performance but it is not the case here. That may be because we have fixed the number of epochs to 100 for all the cases. When learning rate is very small, it takes more time to converge. That might be the reason for this discrepancy.

c. A Few Examples

Image	Question	Target Answer	Predicted Answer
	Where does the ct scan show the hematoma?	left parietal area	in the right parietal
	What does the abdominal ct demonstrate?	a large heterogenous para-aortic mass	a large mass
	What does the ct scan show?	large cyst in the spleen	a cystic mass on
	What does the ct scan show?	multiple liver metastases	multiple cerebral

9. Conclusions & What more?

The final model(with batch norm, drop out=0.3, learning rate =0.001, rmsprop optimization algorithm) produced a BLEU score of 0.393 which is higher as compared to other model variants.

The following are other ideas/techniques that can be used in future to improve the performance of the model:

- Using a image classifier trained on ‘Medical’ images for feature extraction is preferred over using VGGNet as more meaningful feature would have been extracted. Or we could have created our own CNN model for extracting features from the given medical images.
- Deeper LSTMs: Deeper LSTMs in general perform better. One possibility could have been to use stacked LSTMs to improve the BLEU score.
- Pre-trained word-embeddings: Here, we have used an embedding layer and the model learns the weights as part of the training process. Instead, we can use pre-trained word embeddings (using word2vec or Glove) and pass it to the LSTM to get encoding for the question.
- Beam Search: During inference, we have picked the word with highest probability at each time step. Greedy approach may be not be best approach.Beam search can be a good alternative as it tries to maximizes the probability of entire sequence and not a word.
- Attention Mechanism: Attention mechanism allows decoder to attend to different parts of the input at each step of the output generation. Instead of encoding the input into single fixed context vector, we let the model learn what to attend which possibly can better the model.

10. References

- https://www.oreilly.com/learning/caption-this-with-tensorflow
- https://weiminwang.blog/2017/03/20/using-tensorflow-to-build-image-to-text-deep-learning-application/
- https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html
- https://arxiv.org/pdf/1610.02692.pdf