Real Time Prediction System for Visual Question Answering

Tameem Alshomari

[1] Eötvös Loránd University, Budapest, Hungary
[2] {h9whav}@inf.elte.hu

Abstract. Due to the rapid development of deep learning methods for visual recognition, natural language processing, computers could perform various complex and difficult tasks. One of the most important task is to have a computer combining various tools for high-level scene interpretation, such as image captioning and visual question answering.

Keywords: Deep Learning · Convolutional neural network · Recurrent neural network ·

1 Visual Question Answering

1.1 Aata Acquisition

There are many publicly available datasets for VQA system such as CLEVR Dataset from Stanford, FigureQA Dataset from Maluuba , and Microsoft Common Objects in Context (MS COCO) dataset. MS COCO dataset is used to solve our problem.. The dataset is divided into three sets viz. train, validation, and test. The test dataset is not labeled so we cannot use it. So validation dataset will be used for testing purpose and it is not required in the modeling process. The training dataset is again divided into 3 parts viz. questions, annotations(answers), and images. There are around 443757 questions,443757 answers, and 82833 images. We can see that one image has multiple questions and corresponding answers.

2 Model Architectures

2.1 Model 1 (Baseline Model)

The classical method of deep learning for the VQA system involves the following 3 stages.
1.Extract features from question
2.Extract features from image
3.Combine them to generate answer

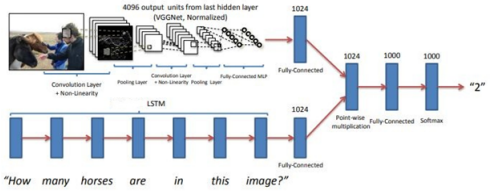

Fig. 1. MODEL 1 (Baseline Model)

2.2 Features Extraction From Questions

The questions are integer encoded with Tokanizer(API) of Keras. They are padded with zeroes so that each encoded sequence will have the same length. Then they are represented as a 300-dim vector using pre-trained GloVe representation. GloVe stands for Global Vectors. It is a vector representation of words. Glove considers local as well global context of words to form a vector. Each question word encoded with 300-dim embedding layer is fed to the LSTM layer. The input vocabulary to embedding layer consists of all the question words seen in the training dataset. An LSTM with two hidden layers is used to obtain 1024-dim embedding for the question.

2.3 Features Extraction From Images

Image features are extracted using transfer learning from pre-trained networks like VGG16, Resnet50 , InceptionV3, etc. Transfer learning makes use of the knowledge gained while solving one problem and transferring it to a different but related problem. We will be using VGG16 for feature extraction. VGG16 is a convolution neural net (CNN) architecture that was used to win ILSVR(Imagenet) competition in 2014. It is trained on dataset containing of about 14 million images.The last layer which is softmax layer is removed from VGG16 and the features from the last hidden layer of VGGNet are used as 4096-dim image embedding. This is followed by fully connected layer with relu non linearity of 1024-dim so that to have question features and image features same shape.

2.4 Combining Features To Generate Answers

The main difference between several approaches is how they combine textual and image features. For example, they can simply combine them using concatenation or using pointwise(element-wise)multiplication of image and answer features and then feed a softmax classifier. They can also use Bayesian models to infer the underlying relationships between the feature distributions of the question, the image, and the answer. We have used pointwise multiplication of question and image features to combine them followed by a fully connected layer of 1000-dim with relu non linearity and softmax layer to obtain distribution over 1000 answers

2.5 Model 2 (BERT Embeddings)

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a neural network-based technique for natural language processing pre-training. It is developed and published by Google. It was designed to discern the context between words. For example, in the phrases "nine to five" and "a quarter to five," the word "to" has two different meanings, which may be obvious to humans but less so to search engines. BERT is designed to distinguish between such nuances to facilitate more relevant results. BERT can train language models based on the entire set of words in a sentence or query (bidirectional training) rather than the traditional way of training on the ordered sequence of words. BERT allows the language model to learn word context based on surrounding words rather than just the word that immediately precedes or follows it. Instead of using GloVe embeddings We will be using BERT embeddings. BERT is used to extract features, namely word and sentence embedding vectors, from questions.

3 Optimizer, Loss Function And Performance Metric Used

3.1 Optimizer

Optimizers update the weight parameters to minimize the loss function. There are various types of optimizers such as SGD, Adagrad, Adadelta, Adam, etc. Various optimizers were tried with different learning rates. Adam optimizer outperforms all the other optimizers.

3.2 Loss Funcion

As part of the optimization algorithm, the error for the current state of the model must be estimated repeatedly. This requires the choice of an error function, conventionally called as a loss function, that can be used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation. There are various types of loss functions to choose from depending on the type of problem. Ours is a multi-class classification problem so we will use categorical cross-entropy as a loss function. Categorical cross-entropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only.

3.3 Performance Metric

Accuracy is a key performance metric that will be used to evaluate the performance of our model. Accuracy is the ratio of total no. of correct predictions to the total no. of input samples.

4 Code Walkthrough

4.1 Loading Data

We are using Google Colaboratory (Colab) platform to write code for this project. First let's import necessary libraries and mount drive so that we can use data directly from the google drive.
We can see that there are total of 443757 questions and the same number of answers. The no. of images are 82433. One image can have a multiple number of questions and answers.

4.2 Transforming Data from Jason to Pandas Dataframe

The question and answer files are in 'Jason' format. We need to convert these into a data frame. Following snippet of code will do that. We are using string formatting to convert image id into an absolute image file path.

4.3 Exploratory Data Analysis

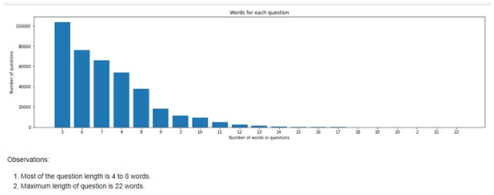

Fig. 2.

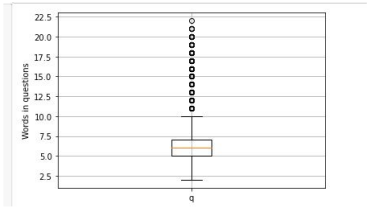

Fig. 3.

Observations: 1. 50 of questions have 5 to 8 words. 2. Most of the questions have less than 10 words although there are few questions having greater than 10 words.

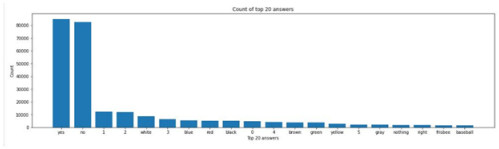

Fig. 4.

Observations: 1. Very large proportion of the answer has yes/no answer. 2. Apart from binary answers most common answers consists of either colors or numbers

```
Total no of binary('yes'/no) answers: 167494
% of binary answers                 : 43.15
Total no of multiple answers        : 220664
% of multiple answers               : 56.85
```

Fig. 5.

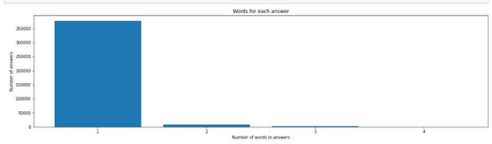

Fig. 6.

Observations: 1. Almost 95 of answers are one word answers although few questions have 2,3 and 4 word answers.

4.4 Data Preparation

We are going to select only those questions and images which have the top 2 answers. Following code, snippet selects only top two answers and corresponding questions and image paths. We will sample 40000 data points from the 167494 data points containing top answers.

4.5 Feature Extraction from Images

Now we will extract image features from the images in the data and store it to disk batchwise in numpy array. Image features are extracted using the last

hidden layer of VGG16 model. vggmodel model is created by removing last softmax layer of pretrained VGG16.We will make set of unique image file paths and pass them to tf.data.dataset as tensor slices.We will decode the images from filepaths and convert them into tensors. We will change the shape of the image to (224,224,3) as VGG16 expects image of that particular size. We will extract features batch-wise and store them to disk in numpy format.

4.6 Train Test Split

Now we will split top data into train and test data using sklearn's train-test-split. The data is divided in such a way that train data has 70 of data and test set has 30 of data.

4.7 Encoding Text Featurest

Now we will encode questions and answers. Questions are text features so they are encoded using Tokenizer api of Keras. Tokenizer API of Keras first split sentence into individual words and then gives unique integer to each word. It is fitted only on train data so that it uses the vocabulary of train data and no data leakage problem occurs. Each sentence has a different length so they are padded with zeroes to make each encoded array have same length. Then they are represented as 300-dim vector using pre-trained GloVe representation. Following two functions do just that.

4.8 Encoding Targets

The answers are encoded using One-Hot-Encoding. One hot encoding is a technique in which categorical variables are represented as a binary vector.

4.9 Creating Dataset

We will create a train and test dataset which will be fed directly to the model. This dataset will load the numpy features stored in the disk while training. we will use tf.data.dataset to create dataset. The dataset is mapped to load features function to load features batchwise. We will zip output and input together so that models can distinguish between input and output. We will batch the complete data with a suitable batch size.

5 Model Architectures

Model Architectures are explained in section 3. We will once again go through models from the code point of view

5.1 Model 1

Model 1 architecture is given in the following code. Input is specified with the expected input shape. We have used batch-wise normalization and dropout wherever required. We have used point wise multiplication to combine image and question features.

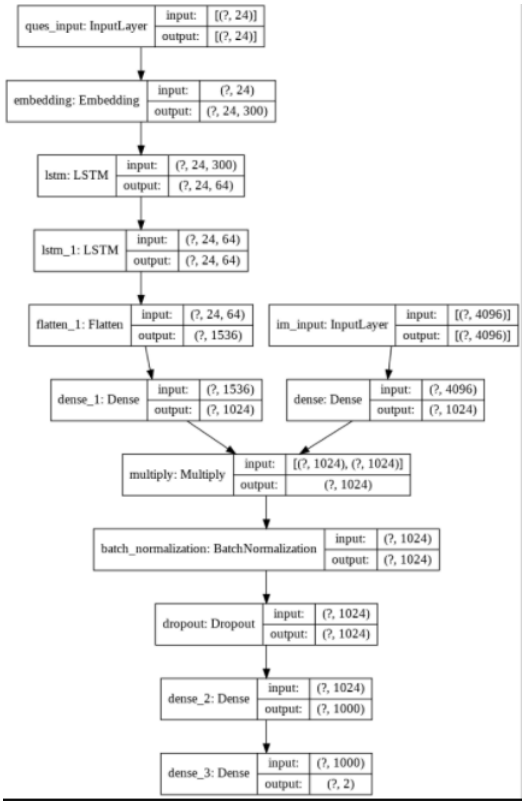

Fig. 7.

There are certain call-backs in Keras, we can use to save our work.Let's take brief overview into it.ModelCheckPoint call-back is used to save best model to the disk.It can also be used to save -weights only.Early-Stopping stops the training if monitored parameter does not improve after certain epochs.TensorBoard is used to plot scalars and histograms.
We will now compile and train our model.

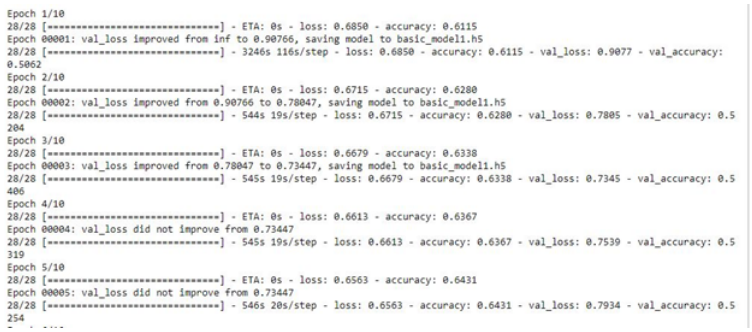

Fig. 8.

5.2 MODEL 2

Model 2 uses BERT embeddings to make vector representations of question features. BERT was originally developed in Tensorflow 1.x so we need to install bert-for-tf2 which is compatible for Tensorflow version 2.0. BERT model "uncased$_L-12_H-768_A-12$" $is fetched from google server which has 12-layer, 768-hidden, 12-heads and 110M parameters. Model checkpoint is also saved.$

BertTokenizer is used to tokenize question text. It splits the words at the end of sentence so that out of vocabulary words can also be handled. We will use BERT model's pretrained weights so we will use trainable=False.

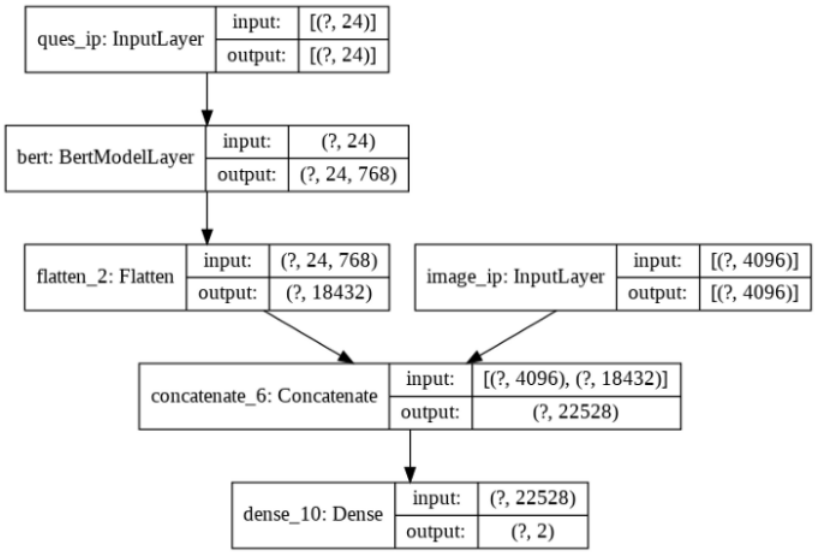

Fig. 9.

```
Epoch 00045: val_loss did not improve from 0.70035
Epoch 46/50
28/28 [==============================] - 37s 1s/step - loss: 0.6777 - accuracy: 0.5898 - val_loss: 0.7212 - val_accuracy: 0.5322

Epoch 00046: val_loss did not improve from 0.70035
Epoch 47/50
28/28 [==============================] - 37s 1s/step - loss: 0.6742 - accuracy: 0.5943 - val_loss: 0.7242 - val_accuracy: 0.5263

Epoch 00047: val_loss did not improve from 0.70035
Epoch 48/50
28/28 [==============================] - 37s 1s/step - loss: 0.6755 - accuracy: 0.5900 - val_loss: 0.7213 - val_accuracy: 0.5297

Epoch 00048: val_loss did not improve from 0.70035
Epoch 49/50
28/28 [==============================] - 37s 1s/step - loss: 0.6855 - accuracy: 0.5859 - val_loss: 0.7221 - val_accuracy: 0.5278

Epoch 00049: val_loss did not improve from 0.70035
Epoch 50/50
28/28 [==============================] - 37s 1s/step - loss: 0.6769 - accuracy: 0.5966 - val_loss: 0.7472 - val_accuracy: 0.5203

Epoch 00050: val_loss did not improve from 0.70035
```

Fig. 10.

5.3 Evaluation

Till now we were just dealing with performance metric to check the performance of the model. But in real world our task will be to get predicted answer as text and not encoded number. We will now create which takes image and question as a input predicts answer as a output.

```
predict_answer("train2014/COCO_train2014_000000458752.jpg", "Is color of pant is white?",'yes')

Question:Is color of pant is white?
Original Answer:yes
Predicted Answer:yes

'yes'
```

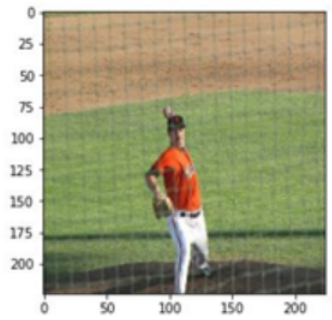

Fig. 11.

We can see that when we give question and image as an input the above function predicts real answer as an output.

5.4 Summary

Model	Model Type	Train loss	Test loss	Train Accuracy	Test Accuracy
Model 1	Baseline Model	0.6679	0.7345	0.6238	0.5406
Model 2	BERT Embeddings Model	0.6934	0.6936	0.5546	0.5412

Fig. 12.

6 Conclusion

1. We developed Visual Question Answering System in this case study. 2. The data was in jason format we converted it into pandas dataframe. 3. There were 443757 questions, 443757 answers, and 82433 images in the dataset. 4. We framed it as a classification problem where given an image and question.related to image we will predict answer from top answers. 5. We applied basic preprocessing steps to maintain data in required form. 6. We explored data using EDA which shows that around 437. As the dataset is huge we sampled 40000 datapoints and top 2 answers. 8. We split the dataset using train-test split with train dataset

having 28000 datapoints and test dataset having 12000 datapoints. 9. We created image features by using last hidden layer of pretrained VGG16 model and stored them to disk. 10. We created train and test dataset using tf.data.dataset which loads the features batchwise. 11. We created 2 model architecutures for our problem viz. baseline model and BERT embeddings model. 12. All the models give encouaraging results with no overfitting. 13. We conclude that BERT model preforms better than other model with 0.69 log loss on train and test data and 54.12

References

1. Jiasen Lu, Jianwei Yang, Dhruv Batra, Devi Parikh Hierarchical Question-Image Co-Attention for Visual Question Answering , https://arxiv.org/abs/1606.00061
2. P. Wang, Q. Wu, C. Shen and A. van den Hengel, "The VQA-Machine: Learning How to Use Existing Vision Algorithms to Answer New Questions," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 3909-3918, doi: 10.1109/CVPR.2017.416.
3. Aishwarya Agrawal , Jiasen Lu , Stanislaw Antol , Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, Devi Parikh VQA: Visual Question Answering.
4. Imane Allaouzi1, Mohamed Ben Ahmed, Badr Benamrou An Encoder-Decoder model for visual question answering in the medical domain VQA-Med of ImageCLEF 2019
5. Yuan-ping Nie Yi Han, Jiu-ming Huang, Bo Jiao Ai-ping Li Attention-based encoder-decoder model for answer selection in question answering 2017.