A REPORT

ON

VISUAL QUESTION ANSWERING

BY

AASHISH AGRAWAL (2015B3A80411G)

ADIT VINAY DESHMUKH (2015B3A70400G)

VARUN B YELIGAR (2015B3A70403G)

Prepared in partial fulfillment of the

Neural Networks and Fuzzy Logic

(BITS F312)

AT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, GOA CAMPUS

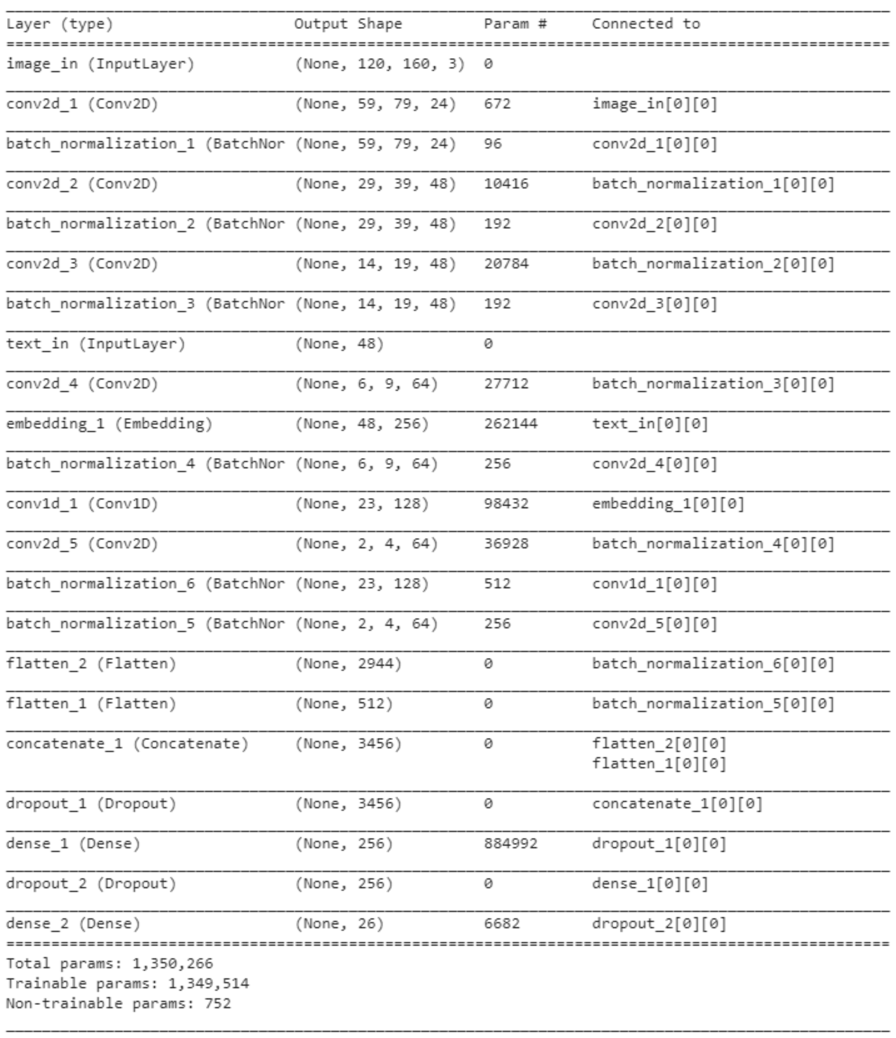
NOVEMBER, 2018

Basic Model

Initially to understand the difficulty and challenges we may face in the task, we designed the most basic of a model which may solve the problem. Since both the image size and the text sequence length were small to begin with, no preprocessing was done on either of them, to get as much information out of them as possible. For constructing the model, we used simple Conv2D layers one after the other which were later flattened, for the purpose of encoding images. Conv1D layers were used to encode the questions text data and was later flattened.

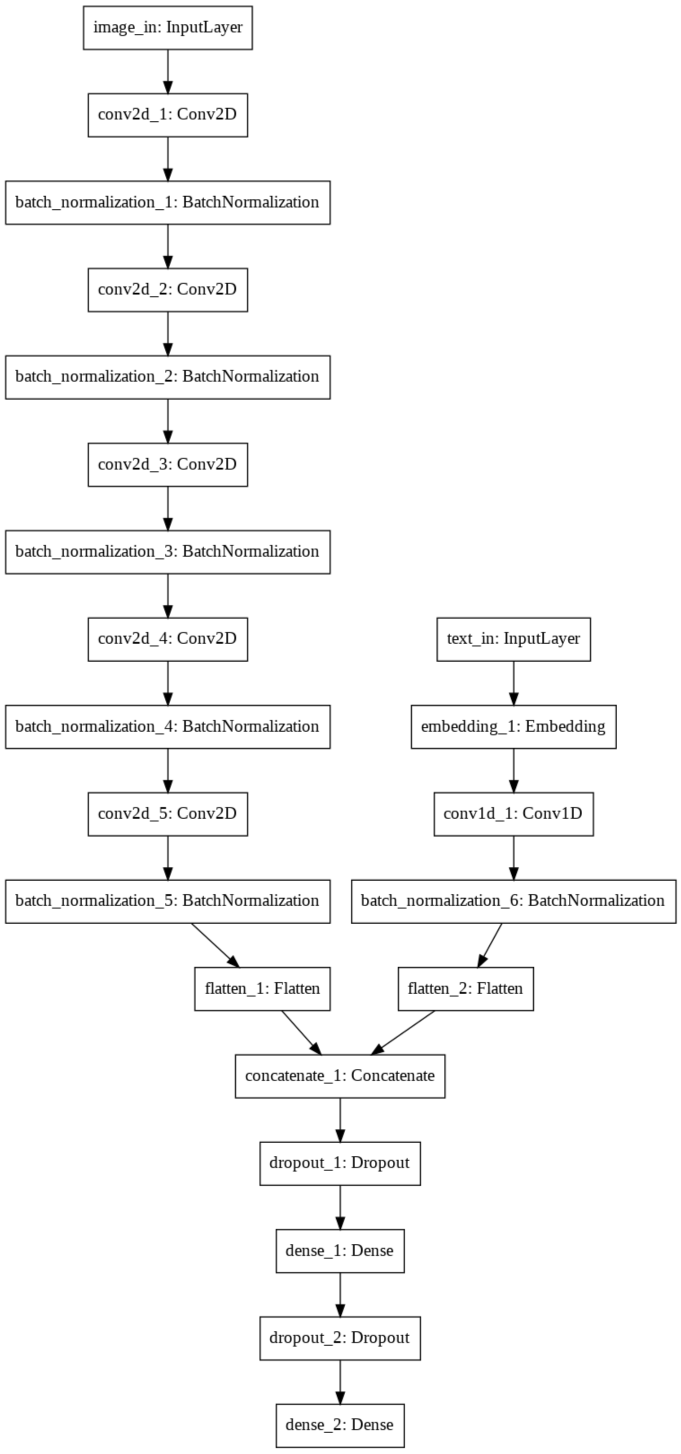
After the encoding of both the text and images data, the encoded vectors thus obtained were concatenated to form a single dense layer. One more Dense layer was appended on top of it and dropouts were used in between them for regularization purposes.

The model summary of the data is presented below:



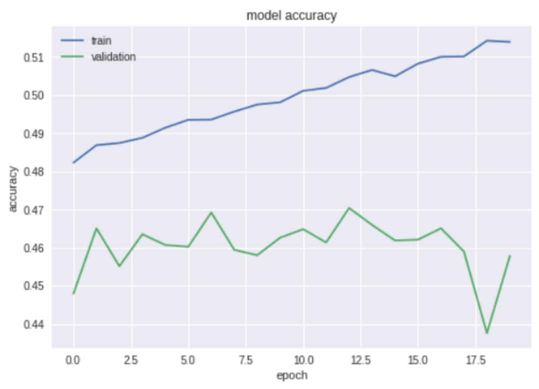
```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
image_in (InputLayer)           (None, 120, 160, 3)  0
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 59, 79, 24)   672         image_in[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 59, 79, 24)   96          conv2d_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 29, 39, 48)   10416       batch_normalization_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 29, 39, 48)   192         conv2d_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 14, 19, 48)   20784       batch_normalization_2[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 14, 19, 48)   192         conv2d_3[0][0]
__________________________________________________________________________________________________
text_in (InputLayer)            (None, 48)           0
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 6, 9, 64)     27712       batch_normalization_3[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 48, 256)      262144      text_in[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 6, 9, 64)     256         conv2d_4[0][0]
__________________________________________________________________________________________________
conv1d_1 (Conv1D)               (None, 23, 128)      98432       embedding_1[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 2, 4, 64)     36928       batch_normalization_4[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 23, 128)      512         conv1d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 2, 4, 64)     256         conv2d_5[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 2944)         0           batch_normalization_6[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 512)          0           batch_normalization_5[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 3456)         0           flatten_2[0][0]
                                                                 flatten_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 3456)         0           concatenate_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          884992      dropout_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 256)          0           dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 26)           6682        dropout_2[0][0]
==================================================================================================
Total params: 1,350,266
Trainable params: 1,349,514
Non-trainable params: 752
__________________________________________________________________________________________________
```

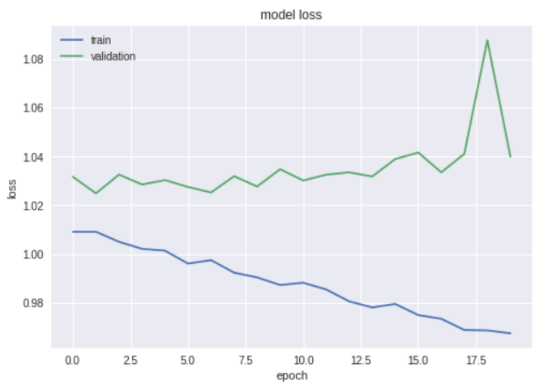
The flow chart of the model is presented below:


image_in: InputLayer
conv2d_1: Conv2D
batch_normalization_1: BatchNormalization
conv2d_2: Conv2D
batch_normalization_2: BatchNormalization
conv2d_3: Conv2D
batch_normalization_3: BatchNormalization
conv2d_4: Conv2D
batch_normalization_4: BatchNormalization
conv2d_5: Conv2D
batch_normalization_5: BatchNormalization
text_in: InputLayer
embedding_1: Embedding
conv1d_1: Conv1D
batch_normalization_6: BatchNormalization
flatten_1: Flatten
flatten_2: Flatten
concatenate_1: Concatenate
dropout_1: Dropout
dense_1: Dense
dropout_2: Dropout
dense_2: Dense

Model thus compiled was trained for 50 epochs, and the resultant files (tokenizer.pickel and model.h5) were stored.

The following graphs plot, accuracy vs epochs and loss vs epochs for both training and validation data. The graph, however, is plotted only for the first 10 epochs, connectivity issues on Google Colab.





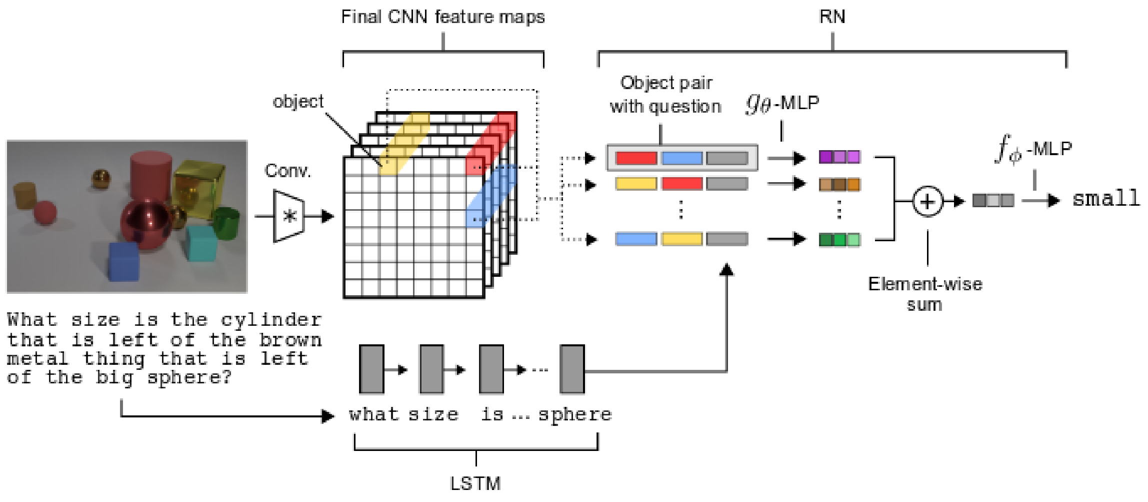
Before we ended up this model, LSTMs were used instead of simple Con1D layers, to encode the text data, however that model was rejected in the initial training phases because of lengthy amount of time taken by it during training and bigger model size, combined with below average results. So, the CNN+LSTM model was discarded in the initial phases only.

Relational Network approach:

The current state of the art result on the CLEVR dataset, of which we have been given a subset, was achieved using relational networks proposed by scientists at Deep Mind. They claimed to have achieved an overall accuracy of 95.5% beating the previous best of 77%. They proposed a new architecture which was built to understand relational dependencies just as convolutional networks are built to understand spatial properties and RNNs are built to understand sequential dependencies. One of the main advantages of the relational model is that it is much smaller in size compared to the previous models and hence is computationally efficient.
The model that we built based on the paper was only 1.8MB in size and achieved accuracy of 45.44% on the validation set. The relational model is smaller but performs better than the other models because it is built to find relational relations and because the complexity in the task is not that the images are complicated but that finding the relations is hard. The relational model addresses this difficulty.

We found that a simpler model which did not use relational networks worked better in practice on the dataset that we had been provided.

The model described in the paper:

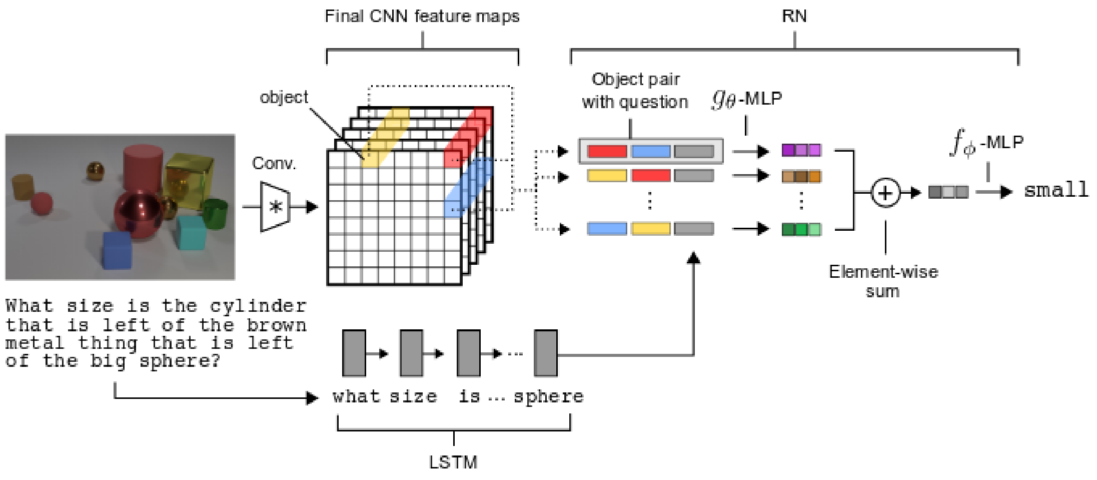

Final CNN feature maps
RN
object
Conv.
Object pair with question
g_θ-MLP
f_ϕ-MLP
small
Element-wise sum
What size is the cylinder that is left of the brown metal thing that is left of the big sphere?
what size is ... sphere
LSTM

Relational model mathematically:

$$\mathrm{RN}(O) = f_\phi\left(\sum_{i,j} g_\theta(o_i, o_j)\right)$$

The summary of the model made in PyTorch:

```
DataParallel(
  (module): RelationNetworks(
    (conv): Sequential(
      (0): Conv2d(3, 24, kernel_size=[3, 3], stride=(2, 2), padding=(1, 1), bias=False)
      (1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): Conv2d(24, 24, kernel_size=[3, 3], stride=(2, 2), padding=(1, 1), bias=False)
      (4): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): ReLU()
      (6): Conv2d(24, 24, kernel_size=[3, 3], stride=(2, 2), padding=(1, 1), bias=False)
      (7): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (8): ReLU()
      (9): Conv2d(24, 24, kernel_size=[3, 3], stride=(2, 2), padding=(1, 1), bias=False)
      (10): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (11): ReLU()
    )
    (embed): Embedding(90, 32)
    (lstm): LSTM(32, 128, batch_first=True)
    (g): Sequential(
      (0): Linear(in_features=180, out_features=256, bias=True)
      (1): ReLU()
      (2): Linear(in_features=256, out_features=256, bias=True)
      (3): ReLU()
      (4): Linear(in_features=256, out_features=256, bias=True)
      (5): ReLU()
      (6): Linear(in_features=256, out_features=256, bias=True)
      (7): ReLU()
    )
    (f): Sequential(
      (0): Linear(in_features=256, out_features=256, bias=True)
      (1): ReLU()
      (2): Linear(in_features=256, out_features=256, bias=True)
      (3): ReLU()
      (4): Dropout(p=0.5)
      (5): Linear(in_features=256, out_features=29, bias=True)
    )
  )
)
```


Final CNN feature maps
RN
object
Conv.
Object pair with question
g_θ-MLP
f_ϕ-MLP
small
Element-wise sum
What size is the cylinder that is left of the brown metal thing that is left of the big sphere?
what size is ... sphere
LSTM

$$\mathrm{RN}(O) = f_\phi\left(\sum_{i,j} g_\theta(o_i, o_j)\right)$$

The current state of the art result on the CLEVR dataset, of which we have been given a subset, was achieved using relational networks proposed by scientists at Deep Mind. They claimed to have achieved an overall accuracy of 95.5% beating the previous best of 77%. They proposed a new architecture which was built to understand relational dependencies just as convolutional networks are built to understand spatial properties and RNNs are built to understand sequential dependencies. One of the main advantages of the relational model is that it is much smaller in size compared to the previous models and hence is computationally efficient.

The model that we built based on the paper was only 1.8MB in size and achieved accuracy of 45.44% on the validation set. The relational model is smaller but performs better than the other models because it is built to find relational relations and because the complexity in the task is not that the images are complicated but that finding the relations is hard. The relational model addresses this difficulty.

We found that a simpler model which did not use relational networks worked better in practice on the dataset that we had been provided.