NEURAL NETWORK AND FUZZY LOGIC

Project Assignment

Visual Question Answering

Nirvan Anjirbag 2016A3PS0711G

Mayank Sharma 2016A8PS0388G

Kishan Kumar Gupta 2016A8PS0406G

An report submitted in

partial fulfilment of the requirement of the course

BITS F312- Neural Network and Fuzzy Logic

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI, KK Birla Goa Campus: 403726

24 November, 2018

Problem Statement:

The dataset provided is a modification of the CLEVR dataset. It consists of a collection of synthesized images and a set of questions associated with each image. Our task is to predict the answers to these questions.

We have been provided with about 15000 images and 135000 questions and answers for training. Most images have about 10 questions associated with them, however, some images don't have any questions associated with them.

The images contain simple 3D objects with each object having one of the 96 property profiles obtained by picking one choice each from the following four *types*:

Shape: Sphere, Cube, Cylinder

Size: Large, Small

Material: (Matte) Rubber, (Shiny) Metal

Color: Gray, Cyan, Blue, Purple, Brown… (8 colors)

Metrics

Accuracy is a common metric for classifiers; it takes into account both true positives and true negatives with equal weight.

accuracy = (true positives + true negatives) / (dataset size)

Algorithms and Techniques

The classifier is a Convolutional Neural Network and LSTM, which is the state-of-the-art algorithm for most image processing tasks, including classification. It needs a large amount of training data compared to other approaches; fortunately, the VQA datasets are big enough. The algorithm outputs an assigned probability for each class; this can be used to reduce the number of false positives using a threshold. (The tradeoff is that this increases the number of false negatives.) The following parameters can be tuned to optimize the classifier:

- Classification threshold
- Training parameters
    - Training length (number of epochs)
    - Batch size (how many images to look at once during a single training step)
    - Solver type (what algorithm to use for learning)
    - Learning rate (how fast to learn; this can be dynamic)
    - Dropout (prevents the model being dominated by a few “neurons”)
    - Momentum (takes the previous learning step into account when calculating the next one)
- Neural network architecture
    - Number of layers
    - Layer types (convolutional, fully-connected, or pooling)
    - Layer parameters (see links above)
- Preprocessing parameters

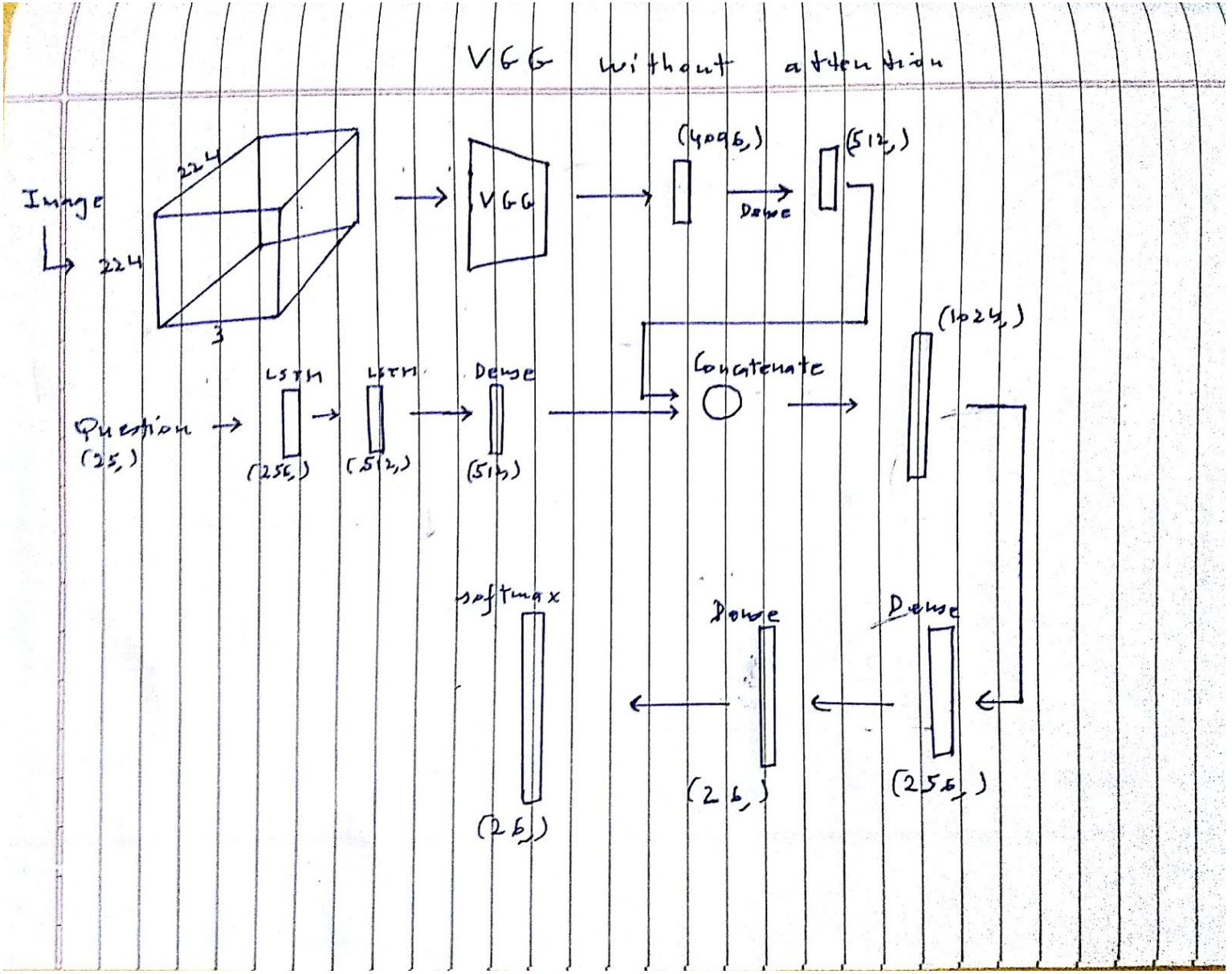
During training, both the training and the validation sets are loaded into the RAM. After that, random batches are selected to be loaded into the GPU memory for processing. The training is done using the Batch gradient descent algorithm (with momentum).

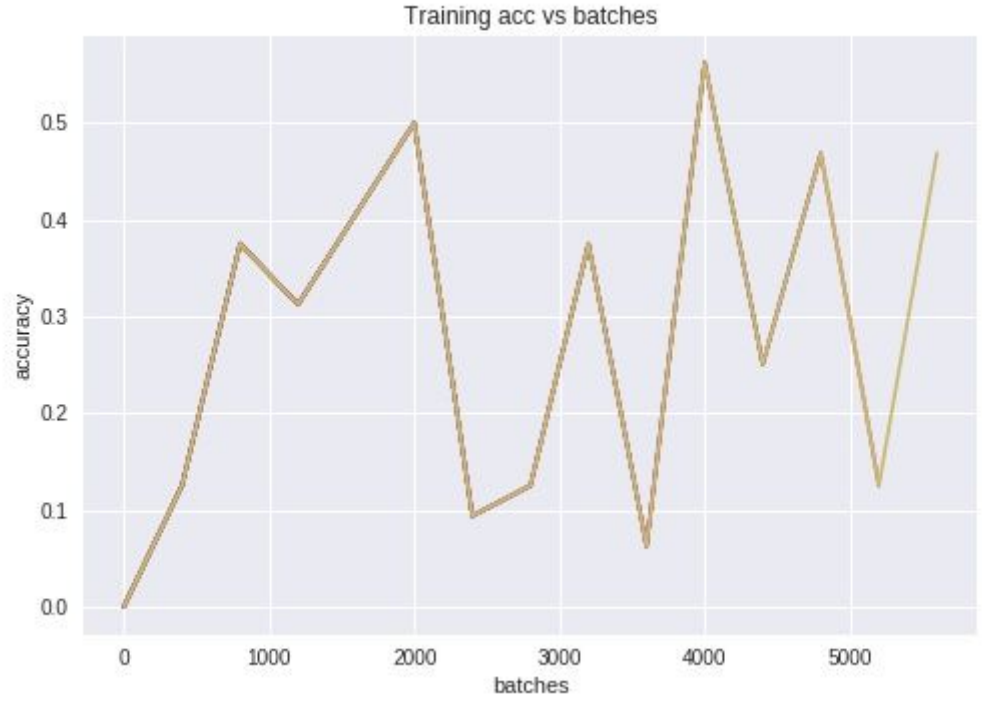
Model 1:

VGG without attention

This model uses a pre-trained VGG network with the last few layers trainable. The image is fed as (224,224,3) and is converted by VGG to a (4096,) vector which is converted by a Dense layer to (512,).

The question is fed as (25,) into the LSTM network and the output question tensor has shape (512,). The output image tensor and the output question tensors are concatenated to form a (1024,) tensor, which is followed by fully-connected layers. The final softmax output is (26,) and is one-hot encoded.


VGG without attention
Image
224
224
3
VGG
(4096,)
Dense
(512,)
Question
(25,)
LSTM
(256,)
LSTM
(512,)
Dense
(512,)
Concatenate
(1024,)
softmax
(26,)
Dense
(26,)
Dense
(256,)


Training acc vs batches
accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0
1000
2000
3000
4000
5000
batches


Training loss vs batches
loss
1.75
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
0
1000
2000
3000
4000
5000
batches

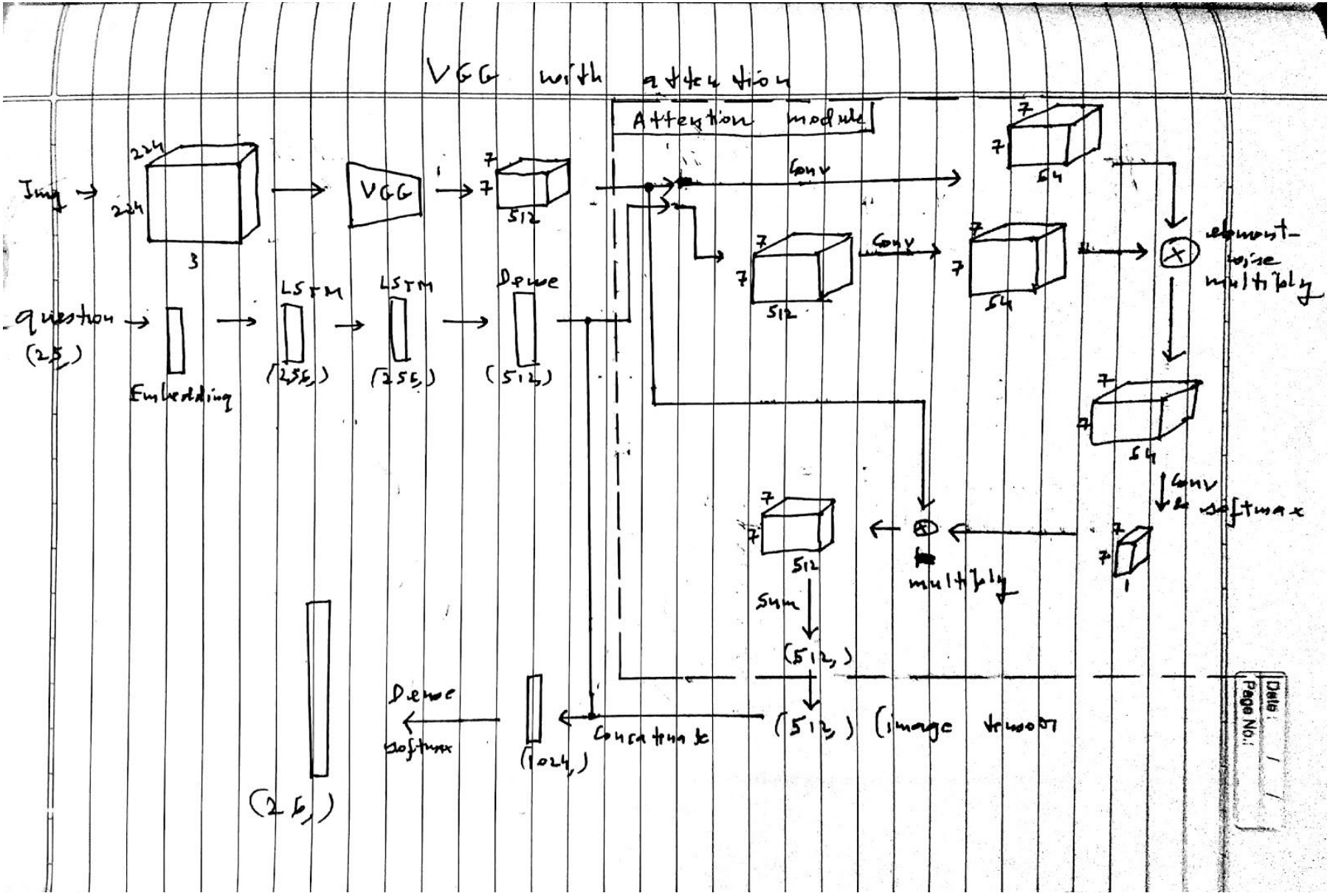
Model 2:
VGG with attention

This model uses a pre-trained VGG network with the last few layers trainable. The image is fed as (224,224,3) and is converted by VGG to a (7,7,512) vector .

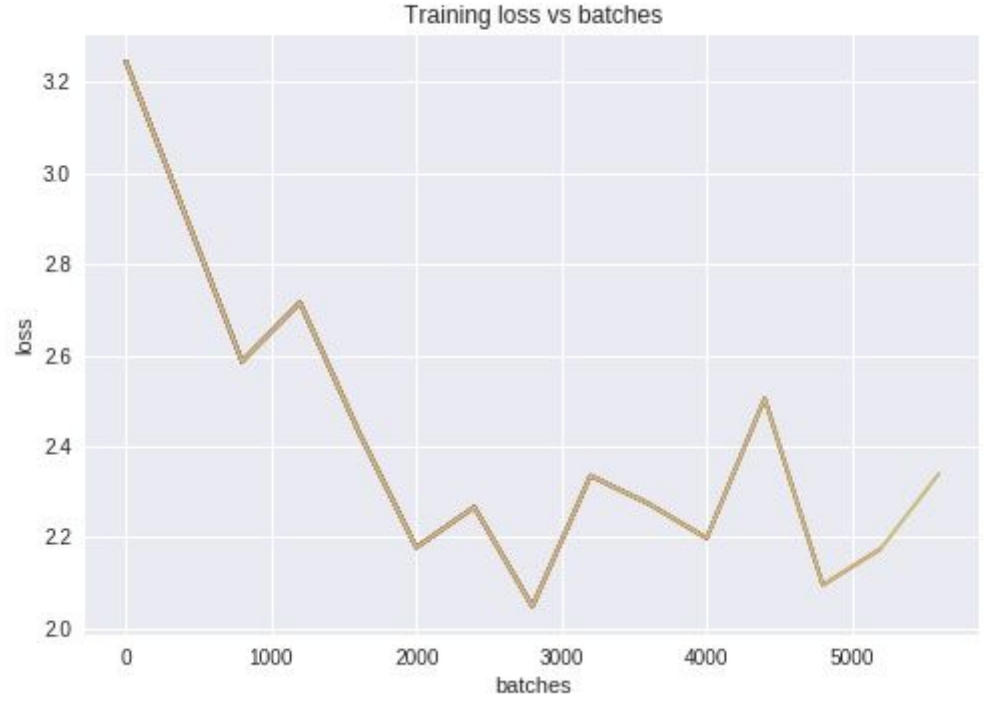
The question is fed as (25,) into the LSTM network and the output question tensor has shape (512,). The output image tensor and the output question tensors are passed to the attention module.

Attention evaluates a weight for each (512,) vector in the vgg output. To do this, every feature vector is multiplied by the question vector. This results in a (7,7,64) vector. This is convoluted to (7,7,1). So now there are 49 weights for the 49 vectors in the VGG output. A weighted sum of the tensors in the vgg output is taken to yield a final image tensor of shape (512,).

The question and image tensors are concatenated to form a (1024,) tensor, which is followed by fully-connected layers. The final softmax output is (26,) and is one-hot encoded.


VGG with attention
Attention module
Img
224
224
3
VGG
7
7
512
Conv
7
7
64
element-wise multiply
question
(25,)
Embedding
LSTM
(256,)
LSTM
(256,)
Dense
(512,)
7
7
512
Conv
7
7
64
7
7
64
Conv softmax
7
7
1
7
7
512
multiply
Sum
(512,)
(512,)
(image tensor)
Dense
softmax
(1024,)
Concatenate
(256,)


Training acc vs batches
accuracy
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0
1000
2000
3000
4000
5000
batches


Training loss vs batches
loss
3.2
3.0
2.8
2.6
2.4
2.2
2.0
0
1000
2000
3000
4000
5000
batches

Model 3:

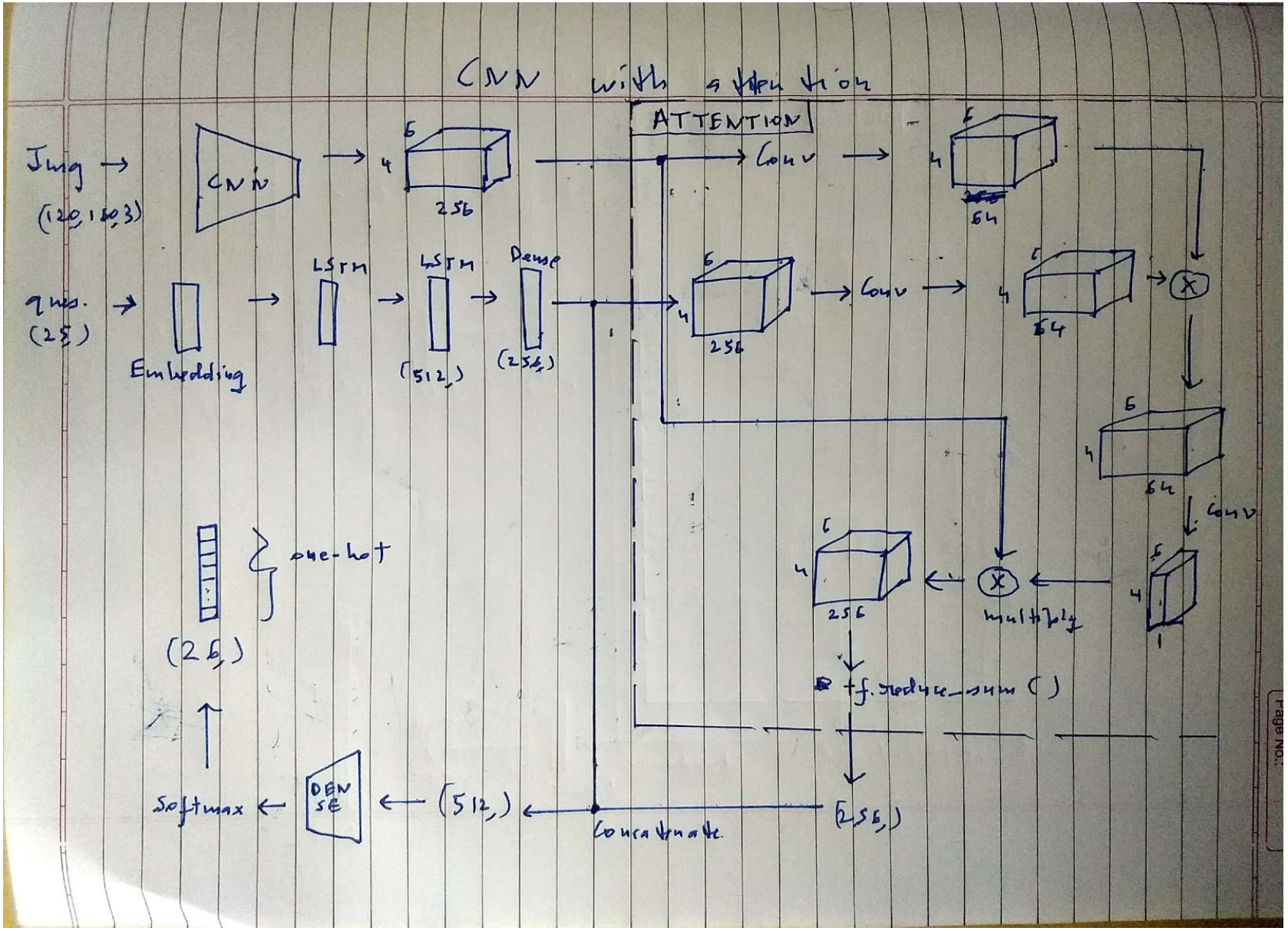
Custom CNN model with attention

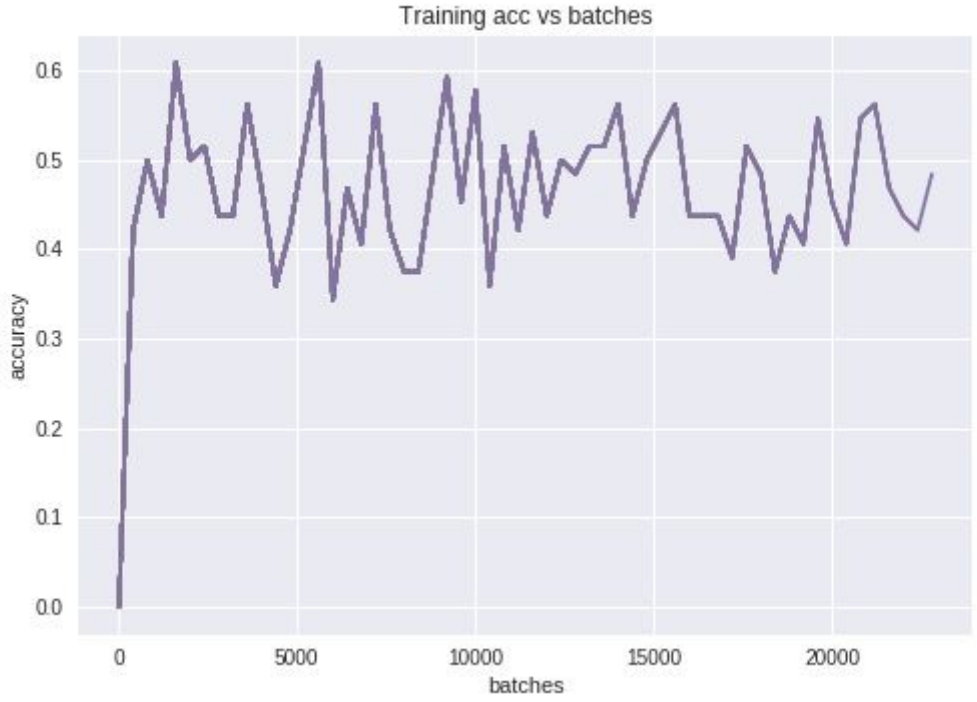
This model uses a CNN network. The image is fed as (120,160,3) and is converted by VGG to a (4,6,256) vector .

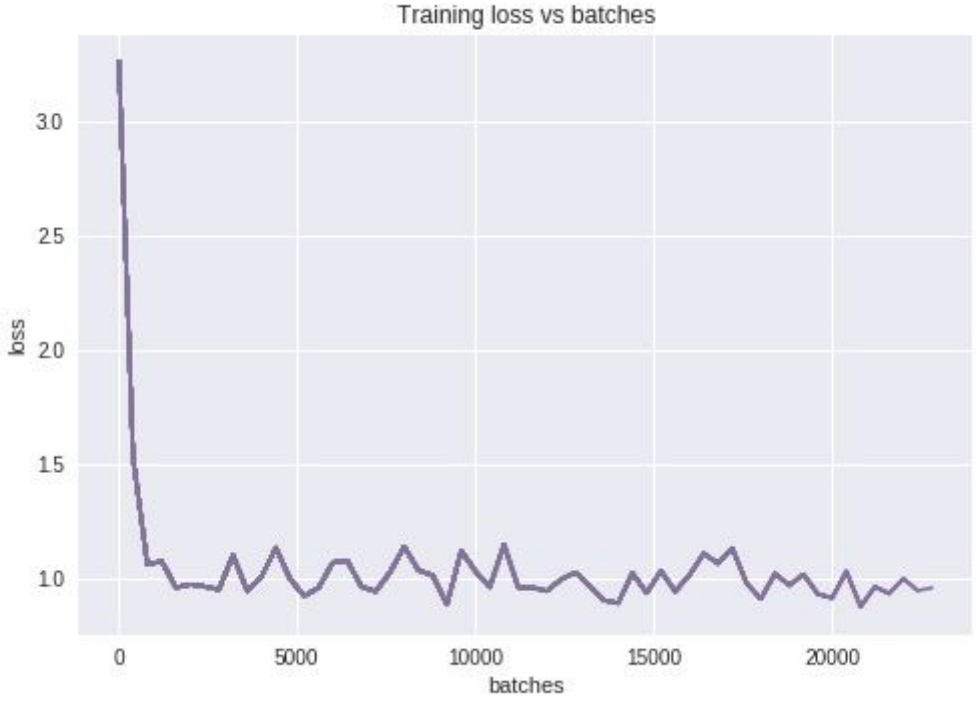
The question is fed as (25,) into the LSTM network and the output question tensor has shape (256,). The output image tensor and the output question tensors are passed to the attention module.

Attention evaluates a weight for each (256,) vector in the vgg output. To do this, every feature vector is multiplied by the question vector. This results in a (4,6,64) vector. This is convoluted to (4,6,1). So now there are 24 weights for the 24 vectors in the CNN output. A weighted sum of the tensors in the vgg output is taken to yield a final image tensor of shape (256,).

The question and image tensors are concatenated to form a (512,) tensor, which is followed by fully-connected layers. The final softmax output is (26,) and is one-hot encoded.


CNN with attention
ATTENTION
Img →
(120, 160, 3)
CNN
8
4
256
Conv
8
4
64
ques.
(25)
Embedding
LSTM
LSTM
(512,)
Dense
(256,)
8
4
256
Conv
8
4
64
8
4
64
Conv
8
4
256
multiply
8
4
1
one-hot
(26,)
tf.reduce_sum( )
Softmax
DENSE
(512,)
Concatenate
(2,56,)





After analysing the graphs we chose custom CNN with attention because the accuracy for it was more stable than others. We obtained train accuracy of 55% and validation accuracy of 49%.