VISUAL QUESTION ANSWERING

Jaineet Shah
Department of Information Systems
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{jaineets@andrew.cmu.edu

Hongyu Mao
Department of Fine Art
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{hongyum@andrew.cmu.edu

Github - https://github.com/jaineet17/Visual-Question-Answering—Applied-Deep-Learning

1 MOTIVATION

In recent years, deep neural networks have made significant progress in computer vision and natural language processing (NLP), leading to highly accurate models for object recognition in images. Despite these advancements, achieving human-level image understanding remains a significant challenge for deep learning models, requiring new approaches that incorporate higher-level reasoning and contextual understanding.

Visual question answering (VQA) is a challenging task that combines computer vision and NLP to generate answers to questions about images. VQA requires a model to recognize objects in the image, comprehend the meaning of the question, and reason about the objects to generate an appropriate answer in natural language. By presenting models with questions about images, VQA can help researchers understand the limitations of current models and develop new approaches that can achieve human-level image understanding.

VQA is challenging because it requires the model to understand the context and semantics of the question, recognize the objects in the image, and reason about their properties and relationships to generate an appropriate answer. Our motivation for exploring multimodal deep learning in VQA is twofold. Firstly, by incorporating both image and natural language inputs, we can improve the accuracy of the VQA model by allowing it to reason about the image in the context of the question. This can enable the model to better understand the relationships between objects in the image and generate more precise and contextually appropriate answers. Secondly, multimodal deep learning in VQA has the potential to provide insights into the cognitive processes involved in human image understanding. By studying how a VQA model combines information from both image and natural language inputs, we can gain a deeper understanding of how humans process visual information and use language to reason about images.

Overall, our research aims to develop more sophisticated and accurate multimodal models for image understanding and contribute to the broader goal of achieving human-level image understanding.

2 SURVEY

There is a lot of research and benchmarks on Visual Question Answering (VQA) from 2015 and it is still ongoing [1]. As Artificial Intelligence (AI) advances and more innovative model architectures are discovered/invented by researchers, the problem of VQA is solved more accurately.

At present, there are mainly four popular datasets - 1) Dataset for Question Answering on Real World Images (DAQUAR). It contains 6794 training and 5674 test question-answer pairs which are based on the NYU-Depth V2 Dataset. 2) Common Objects in Context - Question Answering (COCO-QA). It contains 123,287 images from the COCO dataset with question-answers generated with the help of image captioning. It must be noted that the answers contain only a single word for this dataset. 3) Visual Question Answering (VQA) dataset. It contains all the images from the COCO-QA dataset along with 50,000 abstract cartoon images. For each image, there are three questions. For each

[1] https://paperswithcode.com/task/visual-question-answering

question, there are ten possible answers. This amounts to approximately 760,000 questions and 10 million answers. Also, there are two versions - VQA 1.0 and VQA 2.0 [2]. 4) Compositional Language and Elementary Visual Reasoning (CLEVR). It consists of a training set of 70,000 images and 699,989 questions along with a validation set of 15,000 images and 149,991 questions[3]. Some other less popular VQA datasets are Visual Madlibs, Visual7W, Visual Genome, .[4].

There are several different models with simple to extremely complex architectures, however, the fundamental approach to the problem remains the same. There are four main components in approaching VQA namely Image featurization, Question featurization, Joint feature representation, and Answer generation. Image featurization refers to the process of converting the images to their feature representations for model training. Question featurization refers to the process of converting the questions (text data) to word embeddings for model training. Joint feature representation refers to a technique of combining the feature representations of the images and the question embeddings in an efficient manner that can help the model to understand the input and answer the question. Answer generation refers to the process of utilizing the input image, input question, and joint feature representation to produce an accurate answer as the output[3].

There are plenty of baseline models for VQA trained on different datasets. One of the baseline models is explained in detail in the paper - ”Show, Ask, Attend, and Answer: A Strong Baseline For Visual Question Answering” by Vahid Kazemi and Ali Elqursh. The dataset used in the paper is VQA 1.0. A pre-trained ResNet model is used for image featurization [5].

Long short-term memory (LSTM) network is used for question featurization. LSTM can be described as a sophisticated recurrent neural network (RNN). Conventional RNNs do not perform well with long-term dependencies. However, LSTMs possess more state units commonly known as memory cell which helps to maintain information for longer periods. Therefore, LSTMs in general work better than conventional RNNs as they can handle long-term dependencies in a better manner [6].

The output of the LSTM and ResNet models are concatenated and processed using an attention glimpse layer made of convolutional neural networks (CNNs). Lastly, dense layers and a softmax layer are used for answer generation. Since the authors attempted to generate only one-word answers, they transformed the VQA problem into a classification problem, and therefore, a softmax layer was used at the end for answer generation[4].

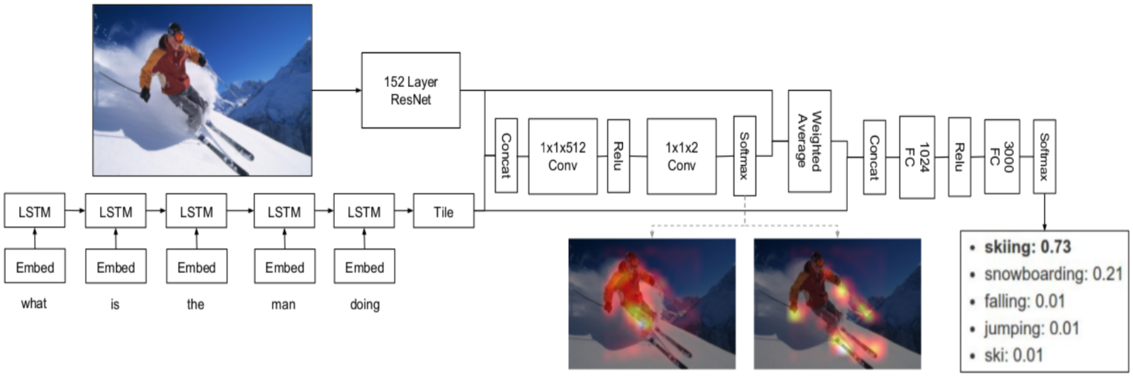


Figure 1: Baseline Model Architecture[4]

From the above figure, we can observe the different components to solve VQA as mentioned above. The dataset used is VQA 1.0 which contains 204,721 images from the COCO dataset. It includes 614,163 questions and 6,141,630 answers.

[2] https://tryolabs.com/blog/2018/03/01/introduction-to-visual-question-answering

[3] https://cs.stanford.edu/people/jcjohns/clevr/

[4] https://blog.paperspace.com/introduction-to-visual-question-answering/

[5] https://arxiv.org/pdf/1704.03162.pdf

[6] https://ashutoshtripathi.com/2021/07/02/what-is-the-main-difference-between-rnn-and-lstm-nlp-rnn-vs-lstm/

Model Specification:- Image Featurization: pre-trained 152 Layer ResNet. Question Featurization: LSTM layer with state size of 1024. Joint Feature Representation: CNN layer of size 512 with 2 attention glimpses. Answer Generation: Dense layer of size 1024 followed by a softmax layer of size 3000.
The above-specified model was trained for 1000 epochs on the training set and an accuracy of 37.16 percent was obtained on the validation set[4].

3 Methodology

As mentioned in the Survey section, the CLEVR dataset is one of the most popular VQA datasets to experiment on. We have tested five types of models with different architectures on the CLEVR dataset.
Model - 1: pre-trained MobileNetV2 + Bi-directional LSTM[7]
Model - 2: VGG-16 Encoder (not pre-trained) + Transformer Encoder
Model - 3: VGG-16 Encoder (not pre-trained) + Bi-directional LSTM
Model - 4: VGG-16 Encoder (not pre-trained) + LSTM + Attention
Model - 5: pre-trained MobileNetV2 + Bi-directional LSTM + Dense

Models: 2-5 are custom models developed by us for this project. We have trained all the models on 20000 images from the training sample of the CLEVR dataset and validated it with 5000 images from the validation sample of the CLEVR dataset. Since the CLEVR dataset has only one-word answers, we have transformed this problem into a classification problem. Hence, there are 99 possible answers or labels for the above-mentioned training sample. The batch size and the number of epochs for all the models are 50 and 10 respectively. Google colab (free version) is used to train the models with a Tesla T4 GPU, Python 3.9.16, and TensorFlow 2.12.0. The questions are tokenized and encoded using the TensorFlow datasets tokenizer and encoder respectively. The shape of the tokens is kept at a fixed length of 50 using padding. A learning rate of 0.001 is used with decay. Sparse categorical cross-entropy and Adam is used as loss function and optimizer respectively.

Model - 1 Specification (pre-trained MobileNetV2 + Bi-directional LSTM)[7]: -
Images were resized to (200,200,3) for this model. Pre-trained MobileNetV2 model on the ImageNet dataset from TensorFlow Keras is used for image featurisation. Three bi-directional LSTM layers with 256, 256, and 512 state units respectively are used for converting questions to features. No attention model is used to combine the image and text features. They are combined with the help of concatenation. Lastly, a Softmax layer with 99 units is used as the classification layer to predict the one-word answer.
The total parameters are 9,333,154. Non - Trainable parameters are 34,112. The average training time per epoch is 190s.

Model - 2 Specification (VGG-16 Encoder (not pre-trained) + Transformer Encoder):-
Images were resized to (224,224,3) for this model. A non-pre-trained VGG-16 architecture is used for image featurisation. A transformer encoder is used for converting questions to features. The transformer encoder consists of 5 layers. Each layer consists of 3 MultiAttentionHeads from TensorFlow Keras followed by an Addition and Layer Normalization layer followed by a Dense layer with dropout regularization and again an Addition and Layer Normalization layer. No attention model is used to combine the image and text features. They are combined with the help of concatenation. Lastly, a Softmax layer with 99 units is used as the classification layer to predict the one-word answer.
The total parameters are 15,057,792. Non - Trainable parameters are 0. The average training time per epoch is 290s.

Model - 3 Specification (VGG-16 Encoder (not pre-trained) + Bi-directional LSTM):-
Images were resized to (224,224,3) for this model. A non-pre-trained VGG-16 architecture is used for image featurisation. Three bi-directional LSTM layers with 256, 256, and 512 state units respectively are used for converting questions to features. No attention model is used to combine

[7] https://www.kaggle.com/code/marcelosabaris/visualquestionanswering

the image and text features. They are combined with the help of concatenation. Lastly, a Softmax layer with 99 units is used as the classification layer to predict the one-word answer.
The total parameters are 21,786,387. Non - Trainable parameters are 0. The average training time per epoch is 310s.

Model - 4 Specification (VGG-16 Encoder (not pre-trained) + LSTM + Attention):-
Images were resized to (224,224,3) for this model. A non-pre-trained VGG-16 architecture is used for image featurisation. One LSTM layer with 256 state units is used for converting questions to features. The image and question features are combined with the help of an attention layer[8]. The attention layer consists of a dense layer with 256 features taking image features as the input. This is followed by a Softmax layer of 1 unit, a multiplication layer with the question features, and a Lambda layer to compute the sum of the weights. The image features and the attention weights are then combined with the help of concatenation. Lastly, a Softmax layer with 99 units is used as the classification layer to predict the one-word answer.
The total parameters are 24,197,540. Non - Trainable parameters are 0. The average training time per epoch is 290s.

Model - 5 Specification (pre-trained MobileNetV2 + Bi-directional LSTM + Dense):-
Images were resized to (200,200,3) for this model. Pre-trained MobileNetV2 model on the ImageNet dataset from TensorFlow Keras is used for image featurisation. Three bi-directional LSTM layers with 256, 256, and 512 state units respectively are used for converting questions to features. No attention model is used to combine the image and text features. They are combined with the help of concatenation. The concatenated layer is followed by two fully connected dense layers with 512 and 256 units respectively. Lastly, a Softmax layer with 99 units is used as the classification layer to predict the one-word answer.
The total parameters are 10,444,195. Non - Trainable parameters are 34,112. The average training time per epoch is 200s.

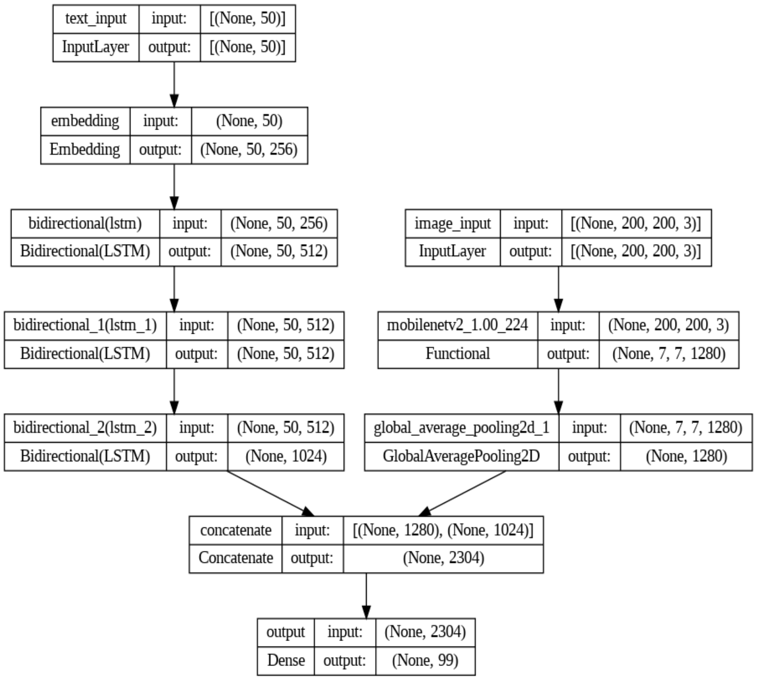

text_input | input: | [(None, 50)]
InputLayer | output: | [(None, 50)]
embedding | input: | (None, 50)
Embedding | output: | (None, 50, 256)
bidirectional(lstm) | input: | (None, 50, 256)
Bidirectional(LSTM) | output: | (None, 50, 512)
image_input | input: | [(None, 200, 200, 3)]
InputLayer | output: | [(None, 200, 200, 3)]
bidirectional_1(lstm_1) | input: | (None, 50, 512)
Bidirectional(LSTM) | output: | (None, 50, 512)
mobilenetv2_1.00_224 | input: | (None, 200, 200, 3)
Functional | output: | (None, 7, 7, 1280)
bidirectional_2(lstm_2) | input: | (None, 50, 512)
Bidirectional(LSTM) | output: | (None, 1024)
global_average_pooling2d_1 | input: | (None, 7, 7, 1280)
GlobalAveragePooling2D | output: | (None, 1280)
concatenate | input: | [(None, 1280), (None, 1024)]
Concatenate | output: | (None, 2304)
output | input: | (None, 2304)
Dense | output: | (None, 99)

Figure 2: Model-1 Architecture

[8] https://blog.dataiku.com/paying-attention-to-text-and-images-for-visual-question-answering

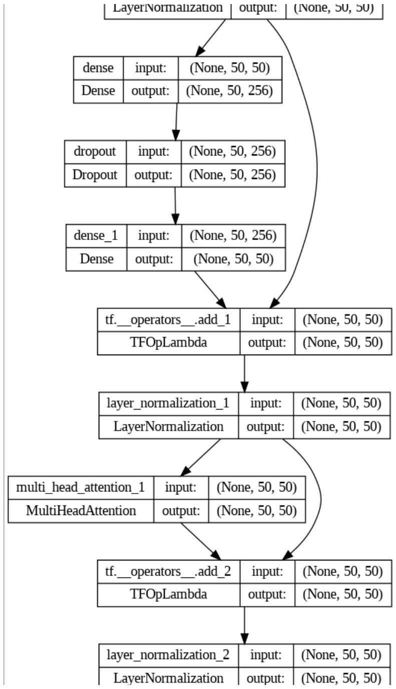

LayerNormalization | output: | (None, 50, 50)
dense | input: | (None, 50, 50)
Dense | output: | (None, 50, 256)
dropout | input: | (None, 50, 256)
Dropout | output: | (None, 50, 256)
dense_1 | input: | (None, 50, 256)
Dense | output: | (None, 50, 50)
tf.__operators__.add_1 | input: | (None, 50, 50)
TFOpLambda | output: | (None, 50, 50)
layer_normalization_1 | input: | (None, 50, 50)
LayerNormalization | output: | (None, 50, 50)
multi_head_attention_1 | input: | (None, 50, 50)
MultiHeadAttention | output: | (None, 50, 50)
tf.__operators__.add_2 | input: | (None, 50, 50)
TFOpLambda | output: | (None, 50, 50)
layer_normalization_2 | input: | (None, 50, 50)
LayerNormalization | output: | (None, 50, 50)

Figure 3: Model-2 Architecture (Partial)

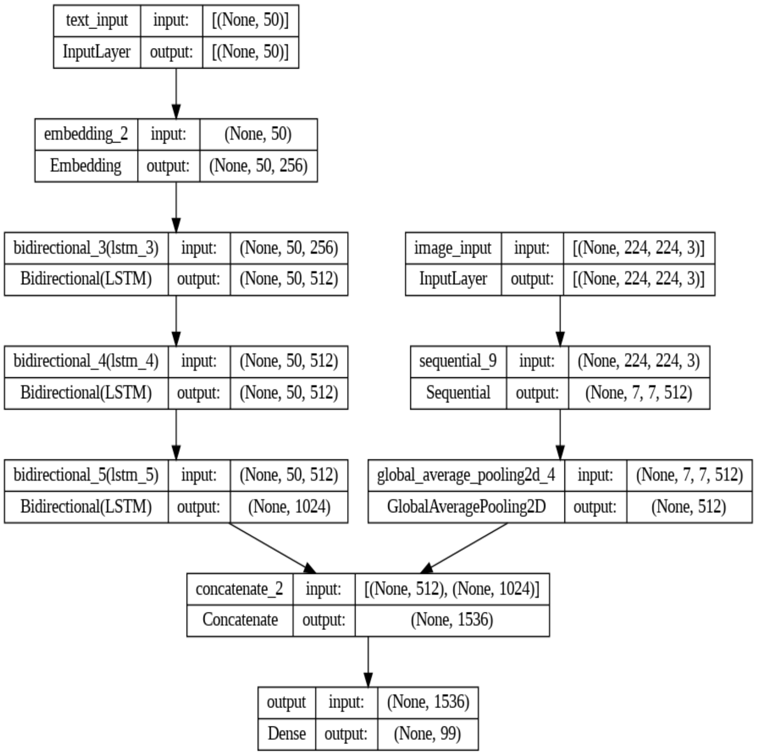

text_input | input: | [(None, 50)]
InputLayer | output: | [(None, 50)]
embedding_2 | input: | (None, 50)
Embedding | output: | (None, 50, 256)
bidirectional_3(lstm_3) | input: | (None, 50, 256)
Bidirectional(LSTM) | output: | (None, 50, 512)
bidirectional_4(lstm_4) | input: | (None, 50, 512)
Bidirectional(LSTM) | output: | (None, 50, 512)
bidirectional_5(lstm_5) | input: | (None, 50, 512)
Bidirectional(LSTM) | output: | (None, 1024)
image_input | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
sequential_9 | input: | (None, 224, 224, 3)
Sequential | output: | (None, 7, 7, 512)
global_average_pooling2d_4 | input: | (None, 7, 7, 512)
GlobalAveragePooling2D | output: | (None, 512)
concatenate_2 | input: | [(None, 512), (None, 1024)]
Concatenate | output: | (None, 1536)
output | input: | (None, 1536)
Dense | output: | (None, 99)

Figure 4: Model-3 Architecture

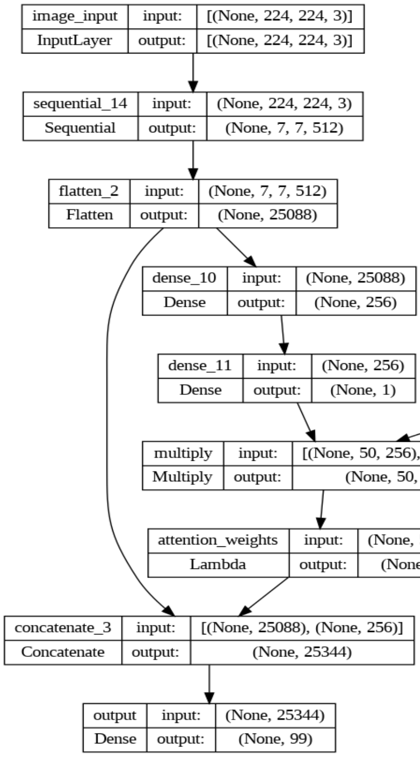

image_input | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
sequential_14 | input: | (None, 224, 224, 3)
Sequential | output: | (None, 7, 7, 512)
flatten_2 | input: | (None, 7, 7, 512)
Flatten | output: | (None, 25088)
text_input | input: | [(None, 50)]
InputLayer | output: | [(None, 50)]
dense_10 | input: | (None, 25088)
Dense | output: | (None, 256)
embedding_3 | input: | (None, 50)
Embedding | output: | (None, 50, 256)
dense_11 | input: | (None, 256)
Dense | output: | (None, 1)
lstm_6 | input: | (None, 50, 256)
LSTM | output: | (None, 50, 256)
multiply | input: | [(None, 50, 256), (None, 1)]
Multiply | output: | (None, 50, 256)
attention_weights | input: | (None, 50, 256)
Lambda | output: | (None, 256)
concatenate_3 | input: | [(None, 25088), (None, 256)]
Concatenate | output: | (None, 25344)
output | input: | (None, 25344)
Dense | output: | (None, 99)

Figure 5: Model-4 Architecture

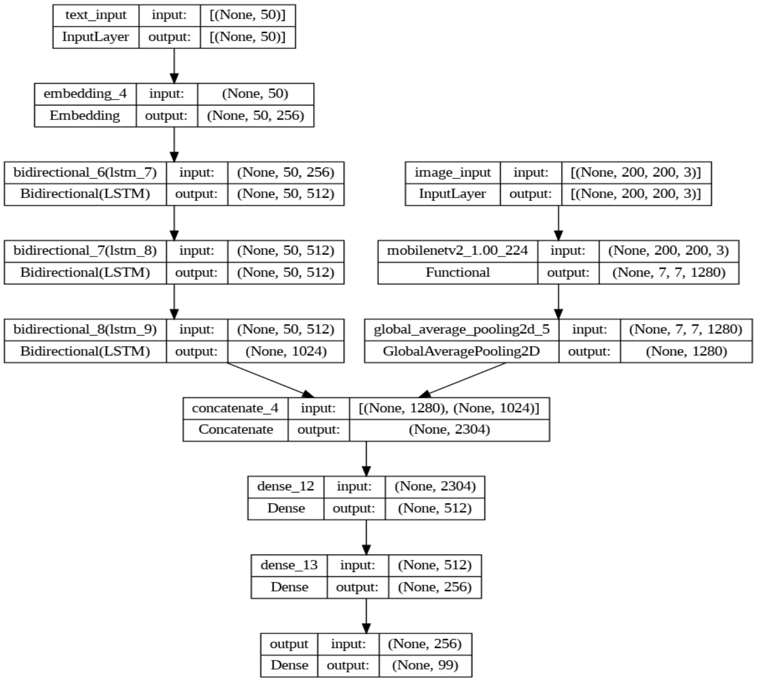
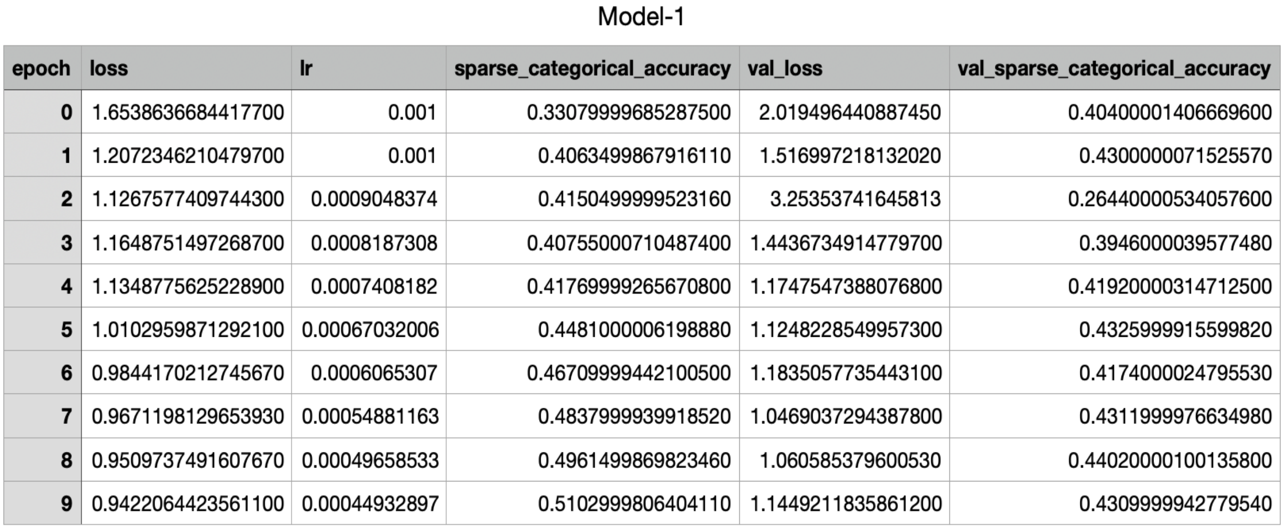

text_input | input: | [(None, 50)]
InputLayer | output: | [(None, 50)]
embedding_4 | input: | (None, 50)
Embedding | output: | (None, 50, 256)
bidirectional_6(lstm_7) | input: | (None, 50, 256)
Bidirectional(LSTM) | output: | (None, 50, 512)
image_input | input: | [(None, 200, 200, 3)]
InputLayer | output: | [(None, 200, 200, 3)]
bidirectional_7(lstm_8) | input: | (None, 50, 512)
Bidirectional(LSTM) | output: | (None, 50, 512)
mobilenetv2_1.00_224 | input: | (None, 200, 200, 3)
Functional | output: | (None, 7, 7, 1280)
bidirectional_8(lstm_9) | input: | (None, 50, 512)
Bidirectional(LSTM) | output: | (None, 1024)
global_average_pooling2d_5 | input: | (None, 7, 7, 1280)
GlobalAveragePooling2D | output: | (None, 1280)
concatenate_4 | input: | [(None, 1280), (None, 1024)]
Concatenate | output: | (None, 2304)
dense_12 | input: | (None, 2304)
Dense | output: | (None, 512)
dense_13 | input: | (None, 512)
Dense | output: | (None, 256)
output | input: | (None, 256)
Dense | output: | (None, 99)

Figure 6: Model-5 Architecture

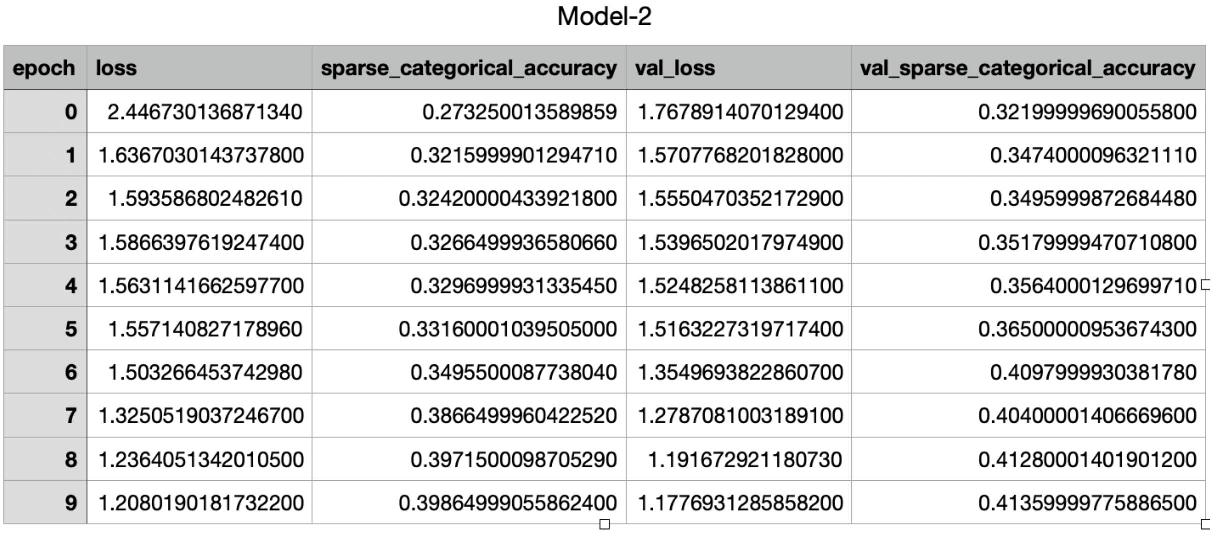
4 RESULTS

From the below graphs, tables, and figures, we can observe that all the models from 1-5 are showing similar performance over 10 epochs of training. However, Model-2, 3 and 4 display slightly more promising trends than 1 and 5 based on the loss and accuracy graphs.



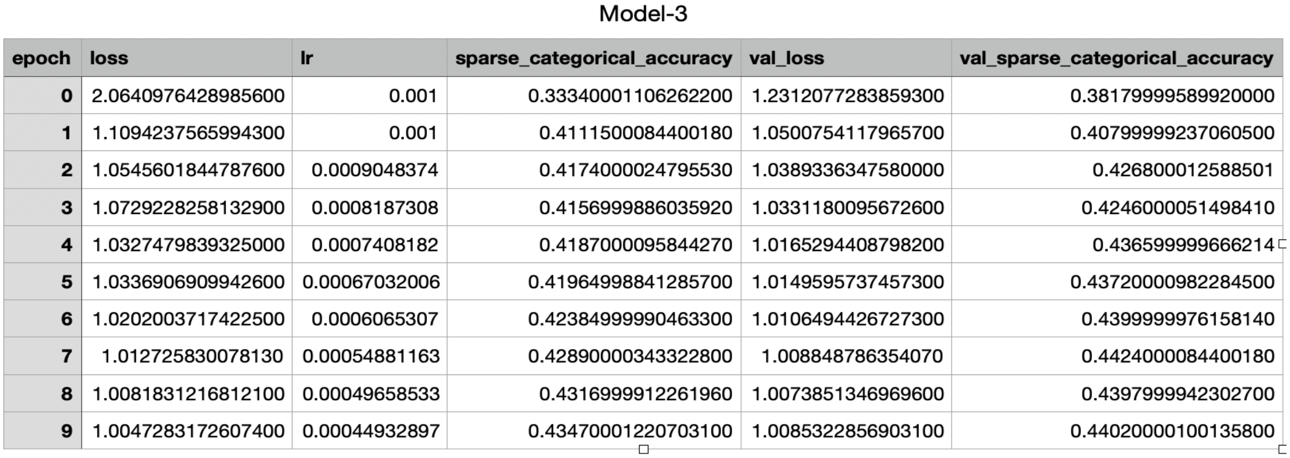
Model-1

epoch	loss	lr	sparse_categorical_accuracy	val_loss	val_sparse_categorical_accuracy
0	1.6538636684417700	0.001	0.33079999685287500	2.019496440887450	0.40400001406669600
1	1.2072346210479700	0.001	0.4063499867916110	1.516997218132020	0.4300000071525570
2	1.1267577409744300	0.0009048374	0.4150499999523160	3.25353741645813	0.26440000534057600
3	1.1648751497268700	0.0008187308	0.40755000710487400	1.4436734914779700	0.3946000039577480
4	1.1348775625228900	0.0007408182	0.41769999265670800	1.1747547388076800	0.41920000314712500
5	1.0102959871292100	0.00067032006	0.4481000006198880	1.1248228549957300	0.4325999915599820
6	0.9844170212745670	0.0006065307	0.46709999442100500	1.1835057735443100	0.4174000024795530
7	0.9671198129653930	0.00054881163	0.4837999939918520	1.0469037294387800	0.4311999976634980
8	0.9509737491607670	0.00049658533	0.4961499869823460	1.060585379600530	0.44020000100135800
9	0.9422064423561100	0.00044932897	0.5102999806404110	1.1449211835861200	0.4309999942779540



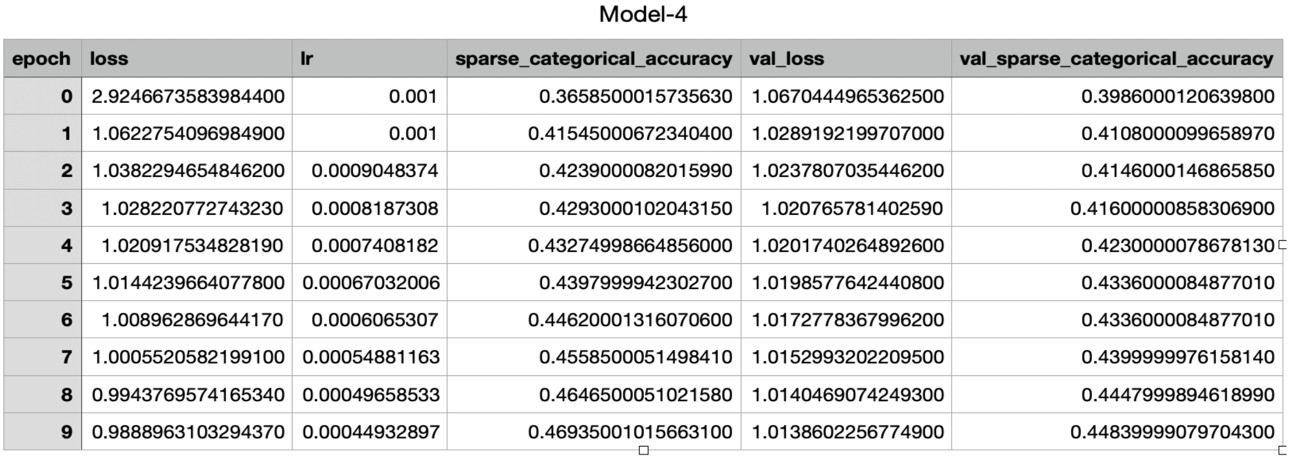
Model-2

epoch	loss	sparse_categorical_accuracy	val_loss	val_sparse_categorical_accuracy
0	2.446730136871340	0.273250013589859	1.7678914070129400	0.32199999690055800
1	1.6367030143737800	0.3215999901294710	1.5707768201828000	0.3474000096321110
2	1.593586802482610	0.32420000433921800	1.5550470352172900	0.3495999872684480
3	1.5866397619247400	0.3266499936580660	1.5396502017974900	0.35179999470710800
4	1.5631141662597700	0.3296999931335450	1.5248258113861100	0.3564000129699710
5	1.557140827178960	0.33160001039505000	1.5163227319717400	0.36500000953674300
6	1.503266453742980	0.3495500087738040	1.3549693822860700	0.4097999930381780
7	1.3250519037246700	0.3866499960422520	1.2787081003189100	0.40400001406669600
8	1.2364051342010500	0.3971500098705290	1.191672921180730	0.41280001401901200
9	1.2080190181732200	0.39864999055862400	1.1776931285858200	0.41359999775886500



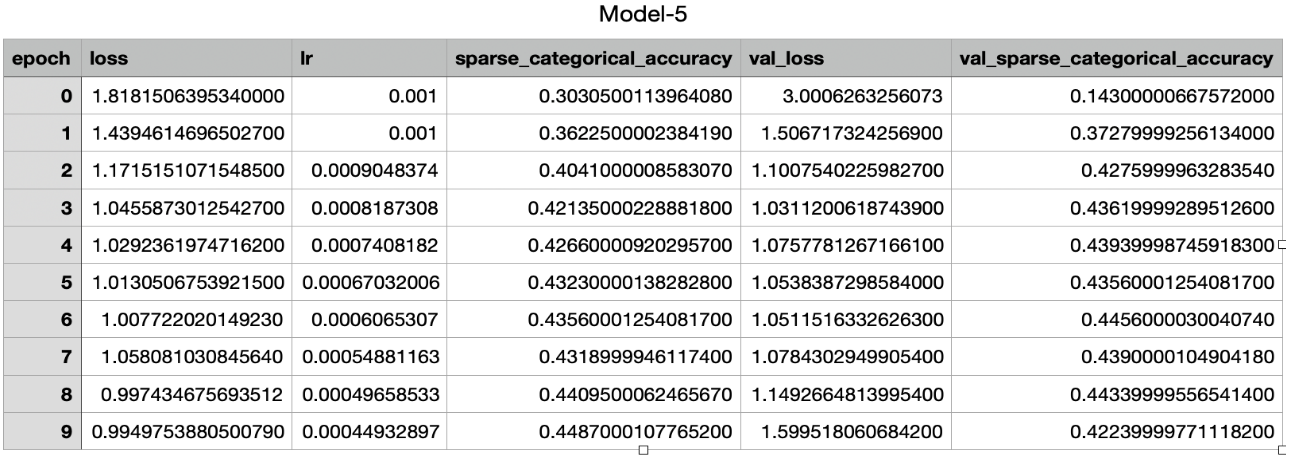
Model-3

epoch	loss	lr	sparse_categorical_accuracy	val_loss	val_sparse_categorical_accuracy
0	2.0640976428985600	0.001	0.33340001106262200	1.2312077283859300	0.38179999589920000
1	1.1094237565994300	0.001	0.4111500084400180	1.0500754117965700	0.40799999237060500
2	1.0545601844787600	0.0009048374	0.4174000024795530	1.0389336347580000	0.426800012588501
3	1.0729228258132900	0.0008187308	0.4156999886035920	1.0331180095672600	0.4246000051498410
4	1.0327479839325000	0.0007408182	0.4187000095844270	1.0165294408798200	0.436599999666214
5	1.0336906909942600	0.00067032006	0.41964998841285700	1.0149595737457300	0.43720000982284500
6	1.0202003717422500	0.0006065307	0.42384999990463300	1.0106494426727300	0.4399999976158140
7	1.012725830078130	0.00054881163	0.42890000343322800	1.008848786354070	0.4424000084400180
8	1.0081831216812100	0.00049658533	0.4316999912261960	1.0073851346969600	0.4397999942302700
9	1.0047283172607400	0.00044932897	0.43470001220703100	1.0085322856903100	0.44020000100135800



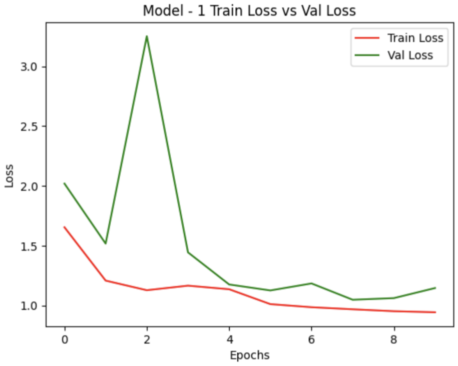
Model-4

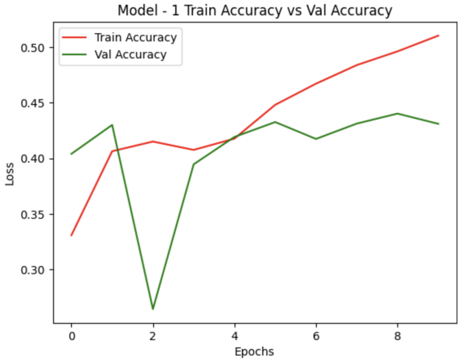
epoch	loss	lr	sparse_categorical_accuracy	val_loss	val_sparse_categorical_accuracy
0	2.9246673583984400	0.001	0.3658500015735630	1.0670444965362500	0.3986000120639800
1	1.0622754096984900	0.001	0.41545000672340400	1.0289192199707000	0.4108000099658970
2	1.0382294654846200	0.0009048374	0.4239000082015990	1.0237807035446200	0.4146000146865850
3	1.028220772743230	0.0008187308	0.4293000102043150	1.020765781402590	0.41600000858306900
4	1.020917534828190	0.0007408182	0.43274998664856000	1.0201740264892600	0.4230000078678130
5	1.0144239664077800	0.00067032006	0.4397999942302700	1.0198577642440800	0.4336000084877010
6	1.008962869644170	0.0006065307	0.44620001316070600	1.0172778367996200	0.4336000084877010
7	1.0005520582199100	0.00054881163	0.4558500051498410	1.0152993202209500	0.4399999976158140
8	0.9943769574165340	0.00049658533	0.4646500051021580	1.0140469074249300	0.4447999894618990
9	0.9888963103294370	0.00044932897	0.46935001015663100	1.0138602256774900	0.44839999079704300

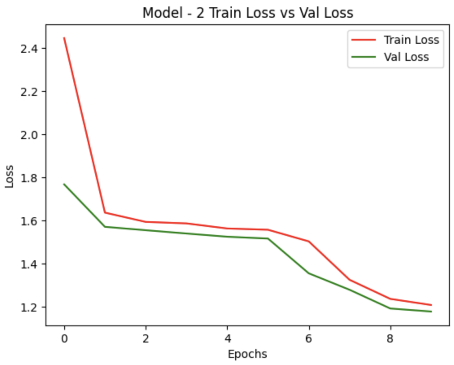


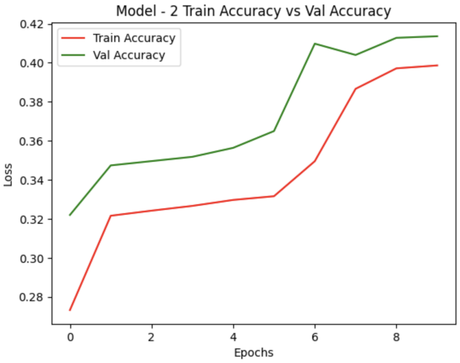
Model-5

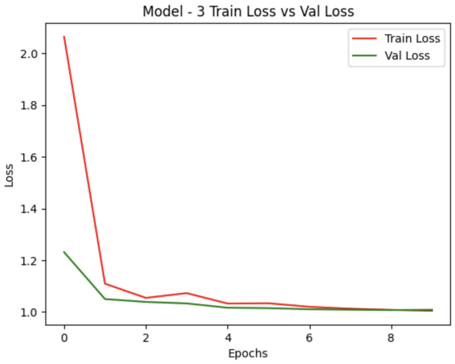
epoch	loss	lr	sparse_categorical_accuracy	val_loss	val_sparse_categorical_accuracy
0	1.8181506395340000	0.001	0.3030500113964080	3.0006263256073	0.14300000667572000
1	1.4394614696502700	0.001	0.3622500002384190	1.506717324256900	0.37279999256134000
2	1.1715151071548500	0.0009048374	0.4041000008583070	1.1007540225982700	0.4275999963283540
3	1.0455873012542700	0.0008187308	0.42135000228881800	1.0311200618743900	0.43619999289512600
4	1.0292361974716200	0.0007408182	0.42660000920295700	1.0757781267166100	0.43939998745918300
5	1.0130506753921500	0.00067032006	0.43230000138282800	1.0538387298584000	0.43560001254081700
6	1.007722020149230	0.0006065307	0.43560001254081700	1.0511516332626300	0.4456000030040740
7	1.058081030845640	0.00054881163	0.4318999946117400	1.0784302949905400	0.4390000104904180
8	0.997434675693512	0.00049658533	0.4409500062465670	1.1492664813995400	0.44339999556541400
9	0.9949753880500790	0.00044932897	0.4487000107765200	1.599518060684200	0.42239999771118200

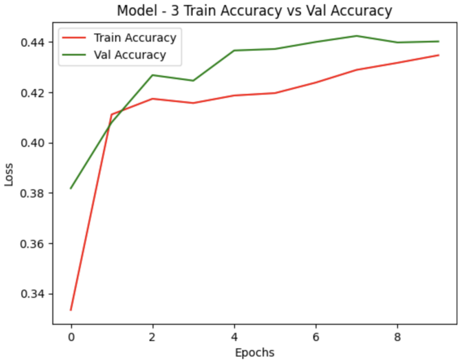

Model - 1 Train Loss vs Val Loss
Train Loss
Val Loss
Loss
3.0
2.5
2.0
1.5
1.0
0
2
4
6
8
Epochs

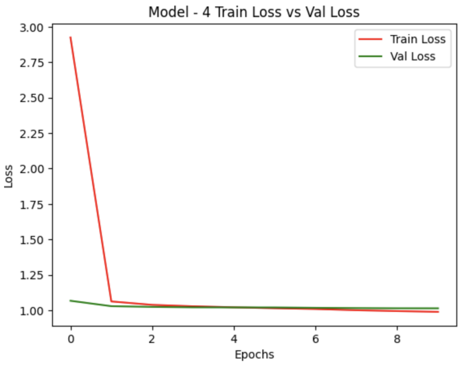

Model - 1 Train Accuracy vs Val Accuracy
Train Accuracy
Val Accuracy
Loss
0.50
0.45
0.40
0.35
0.30
0
2
4
6
8
Epochs

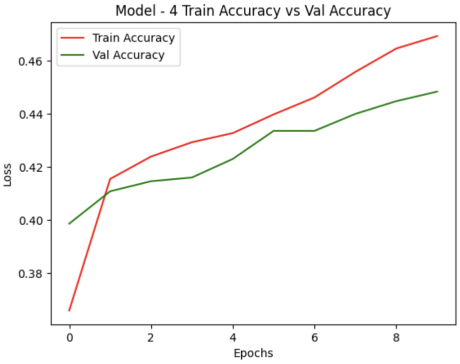

Model - 2 Train Loss vs Val Loss
Train Loss
Val Loss
Loss
2.4
2.2
2.0
1.8
1.6
1.4
1.2
0
2
4
6
8
Epochs

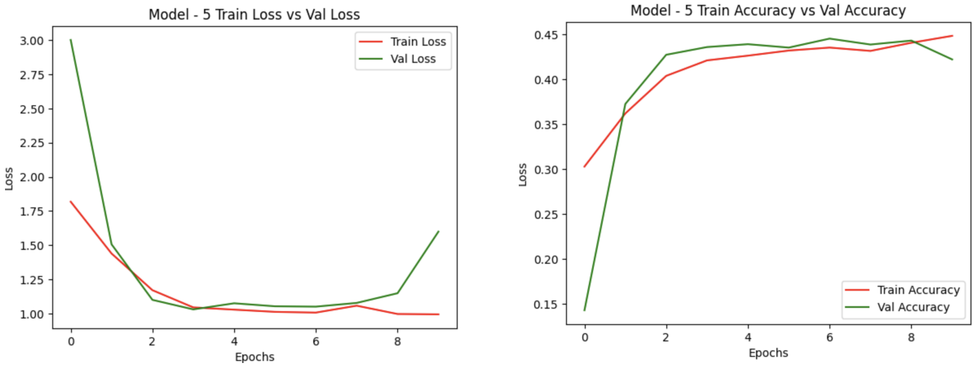

Model - 2 Train Accuracy vs Val Accuracy
Train Accuracy
Val Accuracy
Loss
0.42
0.40
0.38
0.36
0.34
0.32
0.30
0.28
0
2
4
6
8
Epochs

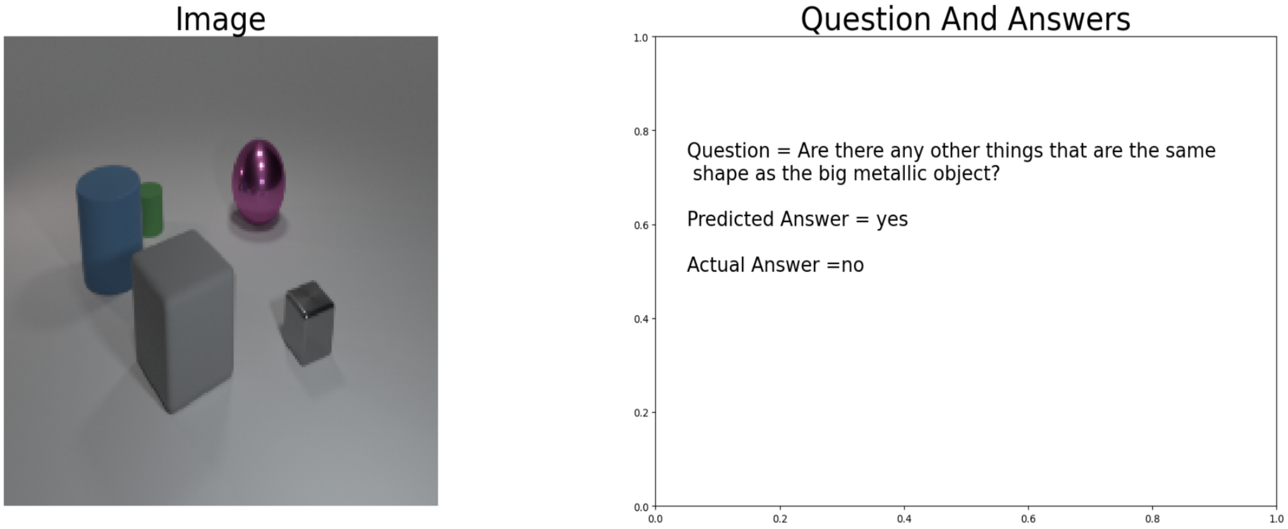

Model - 3 Train Loss vs Val Loss
Train Loss
Val Loss
Loss
2.0
1.8
1.6
1.4
1.2
1.0
0
2
4
6
8
Epochs


Model - 3 Train Accuracy vs Val Accuracy
Train Accuracy
Val Accuracy
Loss
0.44
0.42
0.40
0.38
0.36
0.34
0
2
4
6
8
Epochs


Model - 4 Train Loss vs Val Loss
Train Loss
Val Loss
Loss
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0
2
4
6
8
Epochs


Model - 4 Train Accuracy vs Val Accuracy
Train Accuracy
Val Accuracy
Loss
0.46
0.44
0.42
0.40
0.38
0
2
4
6
8
Epochs



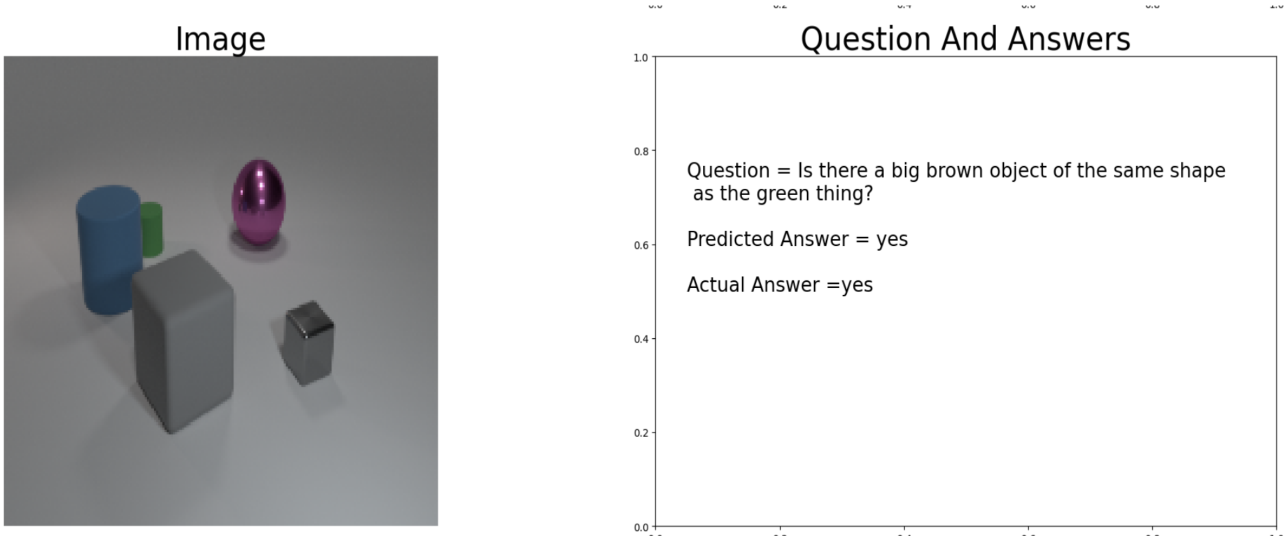


Question = Are there any other things that are the same shape as the big metallic object?

Predicted Answer = yes

Actual Answer =no

Figure 7: Sample Output - 1



Question = Is there a big brown object of the same shape as the green thing?

Predicted Answer = yes

Actual Answer =yes

Figure 8: Sample Output - 2

5 Extension

There are usually various paths to solving any problem in the field of deep learning. Similarly, there are several possibilities to explore while handling VQA.
We have explored a few popular computer vision architectures such as VGG-16 Encoder and MobileNetV2 for image featurization. For question featurization, we have explored some natural language processing (NLP) architectures such as LSTM, Bi-directional LSTM, and transformer encoder. In addition, we have explored some basic attention mechanisms (Model-4 Model-5) to combine the image and question features.
For the future, we plan to properly train (more epochs) and fine-tune our existing models thereby attempting to improve their performance and find their true potential. Later, we also plan to explore more robust techniques to combine image and question feature representations instead of simple concatenation or basic attention and check if it improves performance. Lastly, we will try to evaluate the performance of our trained models on different datasets(DAQAUR, VQA, etc.).

References

"Visual Question Answering (VQA)." Papers With Code. Accessed April 2, 2023. https://paperswithcode.com/task/visual-question-answering.

Couto, Javier. "Introduction to Visual Question Answering: Datasets, Approaches, and Evaluation." Tryolabs. Tryolabs, March 1, 2018. https://tryolabs.com/blog/2018/03/01/introduction-to-visual-question-answering.

Sable, Anuj. "Introduction to Visual Question Answering." Paperspace Blog. Paperspace Blog, April 23, 2021. https://blog.paperspace.com/introduction-to-visual-question-answering/.

Kazemi, Vahid, and Ali Elqursh. "Show, Ask, Attend, and Answer: A Strong Baseline for Visual Question Answering." arXiv.org, April 12, 2017. https://arxiv.org/abs/1704.03162.

Tripathi, Ashutosh, Bijin Abraham says: and Ashutosh Tripathi says: "What Is the Main Difference between RNN and LSTM: NLP: RNN VS LSTM." Data Science Duniya, July 18, 2022. https://ashutoshtripathi.com/2021/07/02/what-is-the-main-difference-between-rnn-and-lstm-nlp-rnn-vs-lstm/.

Marcelosabaris. "Visualquestionanswering." Kaggle. Kaggle, April 5, 2021. https://www.kaggle.com/code/marcelosabaris/visualquestionanswering.

"CLEVR: A Diagnostic Dataset for - Stanford University." Accessed April 16, 2023. https://cs.stanford.edu/people/jcjohns/clevr/.

Phe, François. "Paying Attention to Text and Images for Visual Question Answering." Blog. Accessed April 30, 2023. https://blog.dataiku.com/paying-attention-to-text-and-images-for-visual-question-answering.