# Visual Question Answering with LSTM and Transformer

Tharini Padmagirisan Esha Srivastava
Department of Mathematics, Northeastern University

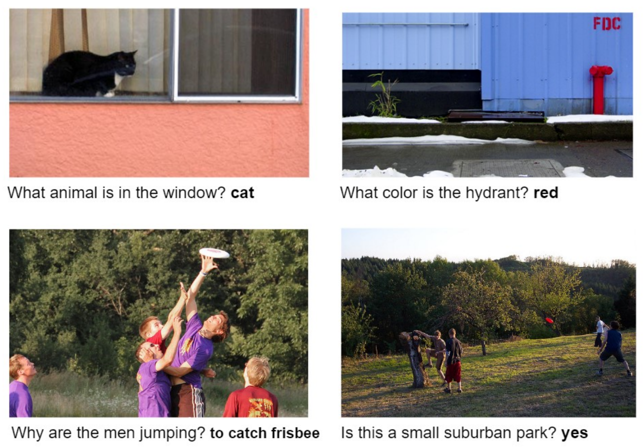
May 22, 2023

## Abstract

*In this study, we built a Visual Question Answering (VQA) System using Neural Networks. A VQA model combines image understanding and natural language interaction. We used a RNN + CNN architecture, the CNN is used to extract image features, and a sequential model is used for natural language interaction, moreover to improve on this baseline model we then used a transformer approach based on LXMERT and studied the results for both the models . Our model has two image feature inputs, at the start and the end of the sentence feature input, with different learned linear transformations. We applied our approach to the standard VQA dataset that contains open-ended questions about the images in the MS-COCO dataset.*

## 1 Introduction

As research in Deep Learning and Neural Networks is progressing, their applications in multi-modal problems such as tasks combining Computer Vision (CV) and Natural Language Processing (NLP) are gaining popularity. Our team is employing Recurrent Neural Networks to solve one such problem, Visual Question Answering (VQA).


What animal is in the window? **cat**
What color is the hydrant? **red**
FDC
Why are the men jumping? **to catch frisbee**
Is this a small suburban park? **yes**

(Some sample images from COCO data set)

A VQA system takes an image and a free-form, open-ended, natural language question about the image as its inputs and produces a natural language answer as the output. This task has many applications such as enhancing navigation for visually-impaired navigators or for intelligence analysts eliciting visual information [1].

For example, to answer the question "Is this a small suburban park?" in the figure with sample images, the model must first understand what is being asked, hence the model needs natural language comprehension. Then, the model also requires common-sense knowledge to understand what it means to refer to a park as small or large. Finally, to determine if the image shown has a suburban park, the model needs visual understanding capabilities in addition to commonsense knowledge. Hence, this problem combines many computer vision sub-tasks such as image labeling and object detection in addition to natural language tasks such as comprehension.

In this project, we built a general visual question-answering model using a neural network based approach. Our implementation is based on the 2-VIS+BLSTM model mentioned in the paper, Exploring Models and Data for Image Question Answering [2]. The Recurrent Neural Networks are primarily used to comprehend the sequential inputs which in our case is the image features concatenated with the question features.Other approach is based on LXMERT: Learning Cross-Modality Encoder Representations from Transformers[6]

## 2 Related work

**Datasets** Many major curated datsets have been released for Visual Question Answering which enable the training and evaluation of VQA systems. A good dataset should be large enough to capture the multiple possibilities within questions and image content and should also allow a fair comparison of performance measures of different approaches. Malinowski and Fritz [8] released the DAtaset for QUestion Answering on Real-world images (DAQUAR), one of the first publicly released datasets for VQA. The DAQUAR dataset is small and contains exclusively indoor scenes, which restricts the variety of questions answerable for each image. There are several other datasets for the VQA tasks. The Visual Madlibs dataset [9] has been collected using fill-in-the-blank templates. The descriptions in this dataset are more detailed and the dataset also contains a multiple choice question answering task for the images.

One of the most important datasets that is a popular benchmark for this task is the VQA dataset. However, the language priors in first version had a significant effect on the accuracy of the models. For example, a simple baseline of always predicting ”yes” as the answer achieved an accuracy of 70.81% on the VQA dataset for the ”yes/no” questions [10]. Hence, Version 2.0 was released in 2017 and has 204,721 COCO images, 1,105,904 questions based on the images, and 11,059,040 ground truth answers. The COCO (Microsoft Common Objects in Context) images were introduced by Lin et al. in Microsoft COCO: Common Objects in Context [11].

**Models for VQA** Generally, a model used for VQA has three components, *Question feature extraction*, *Image feature extraction* and *Combining features for answer prediction*.

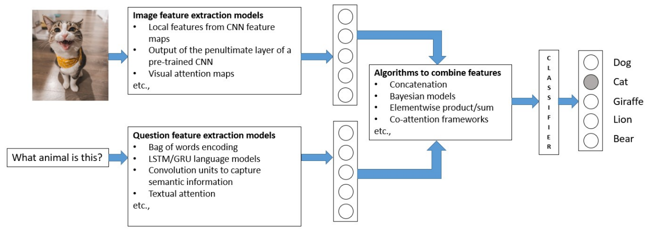


Image feature extraction models
Local features from CNN feature maps
Output of the penultimate layer of a pre-trained CNN
Visual attention maps
etc.,
What animal is this?
Question feature extraction models
Bag of words encoding
LSTM/GRU language models
Convolution units to capture semantic information
Textual attention
etc.,
Algorithms to combine features
Concatenation
Bayesian models
Elementwise product/sum
Co-attention frameworks
etc.,
CLASSIFIER
Dog
Cat
Giraffe
Lion
Bear

(General VQA Architecture)

One of the early approaches proposed was to use CNN to extract image features and to use LSTMs to encode questions. VQA was considered as a classification system where the extracted image and question features were combined and passed through a multi-layer perceptron to predict answers [12]. Later Chen et al. [13] introduced an attention mechanism based architecture for VQA called Attention- based Configurable Convolutional Neural Network (ABC-CNN). More recently, Tan et. al [14] introduced LXMERT, a transformer based architecture that utilizes three encoders, an object relationship encoder, a language encoder, and a cross-modality to learn both intra-modality and cross-modality relationships.

**RNNs for VQA** Malinowski et al also presented an approach [12] that combines semantic parsing and image segmentation. This is one of the first notable approaches in Visual Question Answering. Malinowski and Fritz’s model needs to compute all possible spatial relations in the training images which could be an expensive operation in large datasets. And the accuracy of their model is not high. Gao et al. [15] built an LSTM based model for VQA. Their implementation has two separate LSTMs for questions and answers, while allowing the sharing of the word embedding between the LSTMs in the first and third components out of the total four components. This embedding layer is also shared with the weight matrix of the Softmax layer.

Both Malinowski et al. and Gao et al. used recurrent networks to encode the sentence and output the answer. The approach proposed by Ren et. al also utilizes an LSTM-based architecture to process the sequential question input. But unlike Gao et. al, the task is formulated as a classification problem here. Their methodology is two-fold. On the model side, various forms of neural networks primarily based on LSTMs are developed and they also tried new ways of synthesizing QA pairs from currently available image description datasets on the dataset side.

**Transformers for VQA** LXMERT is built upon several previous works in the fields of computer vision and natural language processing. In terms of language modeling,such as the Transformer, first introduced by [7], BERT (Bidirectional Encoder Representations from Transformers) architecture [3] to encode the input question text. LXMERT also incorporates ideas from previous works on cross-modal learning, such as ViLBERT [4] and VisualBERT [5], which also use transformer-based architectures to combine vision and language modalities. Overall, LXMERT combines and extends several key ideas from previous works to create a powerful cross-modal transformer-based model for VQA tasks.

## 3 Implementation

**Background: Long-Short Term Memory Networks** In recent years, RNNs have been used in a variety of natural language tasks like language modeling, speech recognition, language modeling, translation, and image captioning. Long Short Term Memory networks (LSTMs) are a type of RNN that are capable of learning long-term dependencies. They were introduced by Hochreiter Schmidhuber [16] in 1997. All RNNs have a chain of repeating modules of neural networks. However, LSTMs have a more detailed structure with four neural networks in a repeating module, unlike a standard RNN architecture with a single neural network which only makes predictions by computing a weighted sum of the input signal and applying a non-linear function.

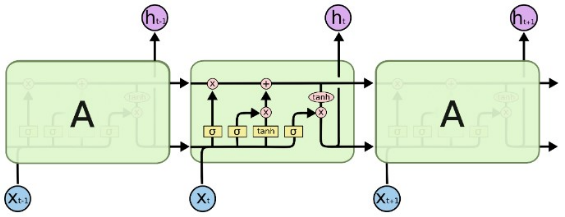
The first layer in an LSTM is a sigmoid layer called the forget gate layer that decides what information is needed anymore. It outputs a number between 0 and 1 for each number in the cell state.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \quad (1)$$

The next layer decides what new information is added to the cell state. This layer has two parts, a sigmoid layer called the input gate layer and a tanh layer that creates a vector of new information to be added to the state.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C) \quad (3)$$



(Layers in an LSTM recurrent unit, Image from Christopher Olah's blog post – Understanding LSTM Networks )

Then, the old cell state, $C_{t-1}$ is updated to the new cell state, $C_t$. We multiply the old state by $f_t$, which forgets the things decided earlier. Then $i_t * \tilde{C}_t$ is added which is the vector of new values.

$$\tilde{C}_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$

Finally, the output is computed. The tanh component pushes the values between -1 and 1, which is then multiplied by the sigmoid output which outputs the decided parts of the current cell state.

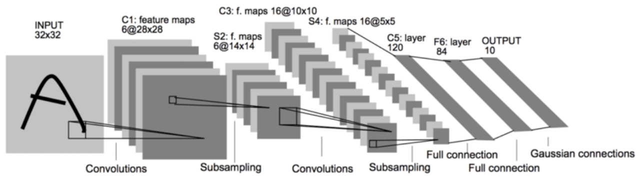
$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * tanh(C_t) \quad (6)$$

Each unit in an LSTM is used to decide whether to retain the previous information and the extent of retention, at each time step. If some early information is determined to be important by the LSTM layers, that information will be carried along through the entire prediction and so the long-term information is captured.

**Background: Conventional Neural Networks** Conventional Neural Networks (CNNs) are feed forward neural networks with convolution layers that replace the direct matrix multiplication operation with convolution operation. The three main layers of a CNN are the Convolution layer, Pooling layer, and Fully-Connected layer.

In the convolution layer, a kernel or a filter which is the feature detector moves across the receptive fields of every training image, checking if a feature, such as edges or facial features is present. This process is known as a convolution. After each convolution operation, a CNN introduces non-linearity by applying a Rectified Linear Unit (ReLU) transformation to the feature map. Pooling layers are the dimensionality reduction layers that reduce the number of input parameters by downsampling. There are two types of pooling operations, Max Pooling and Average Pooling. In a fully-connected layer, each node in the output layer is connected directly to a node in the previous layer. This layer performs classification based on the features extracted through the previous layers. This layer usually uses a softmax activation function to classify based on input features, producing a probability from 0 to 1.



(Image taken from Yann LeCun's 1998 paper)

**Background: Transformers** Transformers are a type of neural network architecture that uses self-attention mechanisms to process sequential data. Let $x_1, x_2, \ldots, x_n$ be an input sequence of length $n$, and let $\mathbf{x}i$ be the embedding of the $i$-th token. The key idea behind the self-attention mechanism is to compute a weighted sum of all the embeddings based on their similarities to each other. This is achieved by computing the attention weights $wi, j$ between all pairs of tokens $(i, j)$ as follows:

$$w_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{n} \exp(e_{i,k})},$$

where $e_{i,j} = \mathbf{x}_i \mathbf{W}_Q (\mathbf{x}_j \mathbf{W}_K)^T$ is a measure of similarity between the $i$-th and $j$-th tokens, $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are learnable matrices used to project the embeddings into a query, key, and value space, respectively.

Once the attention weights are computed, the output for each token is obtained as a weighted sum of the values:

$$\mathbf{y}i = \sum j = 1^n w_{i,j} \mathbf{x}_j \mathbf{W}_V.$$

The Transformer architecture consists of a stack of multiple self-attention layers, each of which is followed by a feed-forward neural network layer. The output of each self-attention layer is passed through a layer normalization and residual connection before being fed into the feed-forward layer.

The original Transformer paper **??** also introduced several modifications to the basic self-attention mechanism, including multi-head attention, which allows the model to attend to different parts of the input sequence simultaneously, and positional encodings, which are added to the embeddings to preserve the order of the tokens.

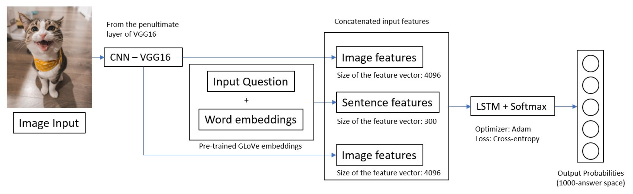
Overall, the Transformer architecture has proven to be highly effective in processing sequential data, particularly in natural language processing tasks. It has been used to achieve state-of-the-art results in a wide range of NLP benchmarks and has become a fundamental tool in the NLP community.

In this project,For the first model we have used CNNs to extract features from the images. Deeper networks provide a richer representation of images with features extracted from multiple layers. So we have used a VGGNet with 16 layers to extract the image features. Since these deep CNNs are hard to train, pre-trained weights with ImageNet data are used in our implementation. For our second model we used LXMERT , which uses pre-trained weights from Hugging face API and used it to train and test on VQA dataset.

## 4 Methodology

### 4.1 LSTM for VQA

The implementation is similar to the 2-VIS+BLSTM model described in Ren et. al [2]. Here the input image is treated as the first and last words of the input question. This idea of treating the image as a word was originally implemented in the caption generation work done by Vinyals et al. [17]. The 2-VIS+BLSTM model proposed in the paper uses bi-directional LSTMs, i.e., LSTMs going in both forward and backward directions. Both LSTMs output to the softmax layer at the last timestep. Our implementation however does not use bi-directional LSTMs, only forward LSTMs are used.


From the penultimate layer of VGG16
Image Input
CNN – VGG16
Input Question
+
Word embeddings
Pre-trained GLoVe embeddings
Concatenated input features
Image features
Size of the feature vector: 4096
Sentence features
Size of the feature vector: 300
Image features
Size of the feature vector: 4096
LSTM + Softmax
Optimizer: Adam
Loss: Cross-entropy
Output Probabilities
(1000-answer space)

(Implemented Architecture)

The last hidden layer of the 16-layer Oxford VGG Conv Net trained on ImageNet Challenge dataset is used to get the visual embeddings. The CNN part of the model is not trained. The image is then treated as the first and last words of the sentence. Similar to DeViSE [18], a linear or affine transformation is used to map 4096 dimension image feature vectors to a 300-dimensional vector that matches the dimension of the word embeddings. The outputs of the LSTMs are then fed into a softmax layer at the last timestep to generate natural language answers.

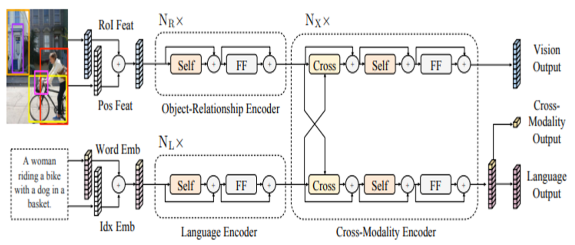
To get the image features from VGG CNNs, we used the pre-computed MS COCO features made available by the authors of Deep Visual-Semantic Alignments for Generating Image Descriptions [19]. To generate the embedding matrix and word indices, we used the pre-trained GloVe word vectors made available by Pennington et. al [20]. Linear transformation is used to map 4096 dimension image feature vectors to a 300-dimensional vector that matches the dimension of the word embeddings.

Our current model is built on Keras functional API. We concatenated sequential layers to get the features of our sequential input with the images as the first and last words of the sentence. This is concatenated with an LSTM layer to extract the features of this input sequence. This is followed by a fully connected layer to interpret the LSTM output with softmax activation for predicting probabilities. For VGG, we used an instance of the Keras model VGG16 instantiated with the pre-trained ImageNet weights. Since training a CNN on a large number of images is a time and resource intensive process, we used the pre-trained weights available on Keras for VGG16. We used 180,000 images for training and 100,000 images for validation from the VQA v2 dataset. For a batch size of 100 images, our Keras implementation takes about 20 minutes per epoch on NVIDIA A100 (40GB) GPU.

### 4.2 LXMERT for VQA

The original implementation of LXMERT was done using the PyTorch deep learning framework. However, it is also possible to implement LXMERT using the Hugging Face Transformers API, which provides a convenient interface for working with pre trained transformer models. To implement LXMERT using the Hugging Face Transformers API, we first loaded the pre-trained LXMERT model and its tokenizer into memory using the 'LxmertForQuestionAnswering' and 'LxmertTokenizer' classes from the 'transformers' module.

Once the model and tokenizer were loaded, we used them to process textual and visual inputs. For instance, to answer a visual question, we would tokenize the question and image using the LXMERT tokenizer and pass the resulting tokens to the LXMERT model for prediction. The model would then use its pre-trained weights to combine the textual and visual inputs and produce an answer to the question.



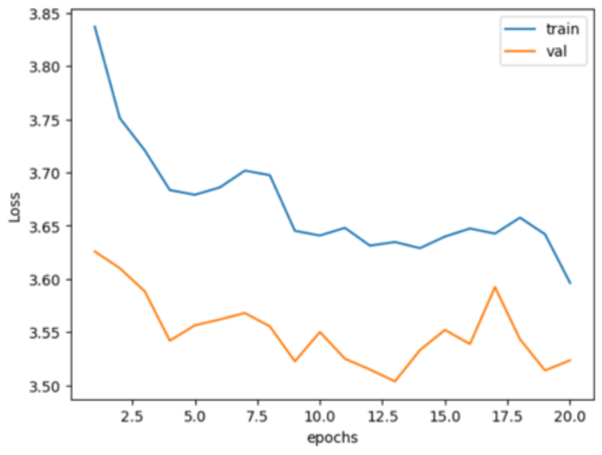
(Implemented Architecture)

We had 64 GB of RAM for this task, the pre-trained LXMERT model and its tokenizer was able to fit into the memory, allowing for efficient processing of multi-modal inputs. However, it's worth noting that running the LXMERT model on large data sets can still be computationally intensive and requires additional hardware resources such as GPUs or TPUs to achieve reasonable performance.Our estimated time to train the whole model was 166 hours and above but due to limitation of time and computational resources we trained it for 10 epochs, each epoch took 2 hours at the least.

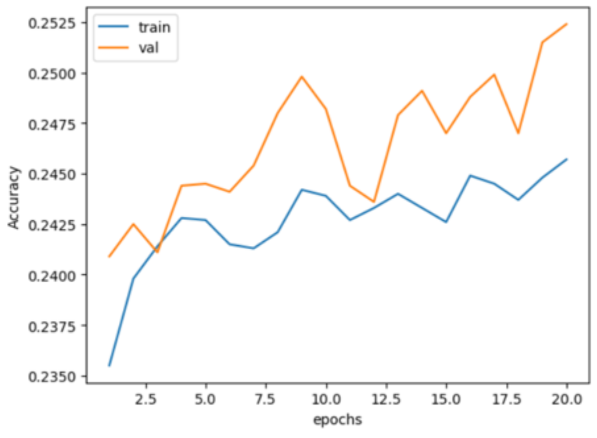
## 5 Results

#### 5.0.1 Accuracy

We have currently trained our model for 20 epochs. Plots of the training accuracy, the validation accuracy and the training loss and the validation loss are shown below,



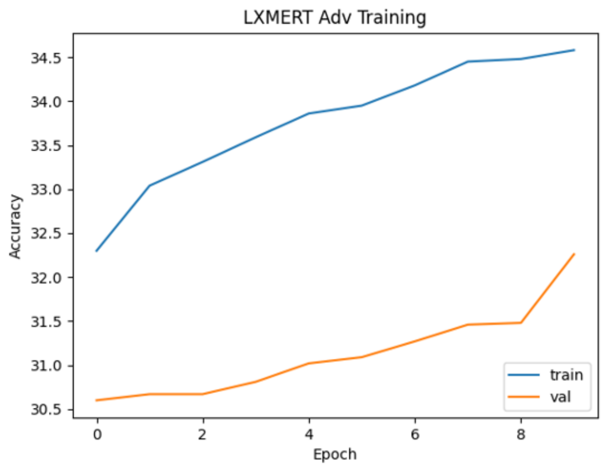
(Model Loss)



(Model Accuracy)

The LSTM model seems to be underfitting and so continuing the training for more epochs would improve the results.

For VQA our accuracy is show below with Adversarial Training.



(Model Accuracy)

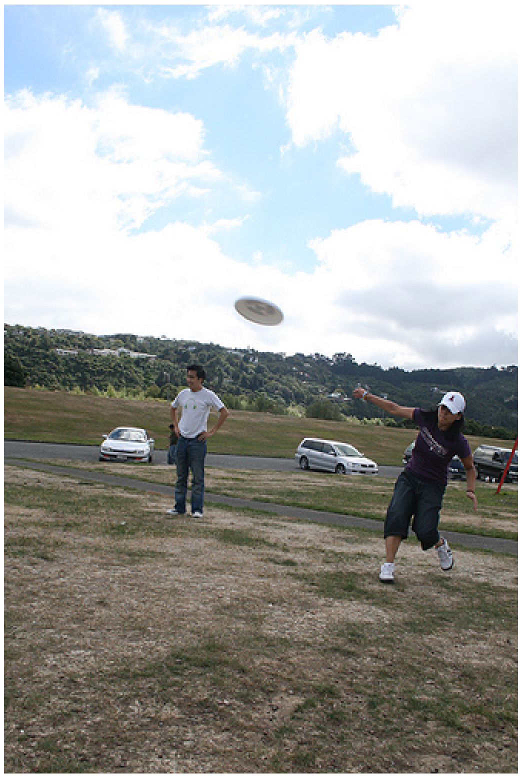
#### 5.0.2 Sample Predictions

Here are some questions and sample responses predicted by our model

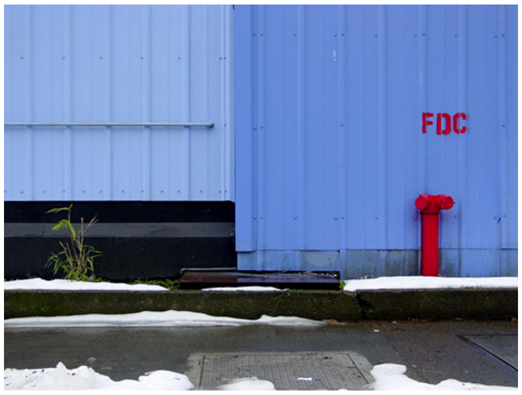
Question: What animal is this?

Ground Truth: Dog

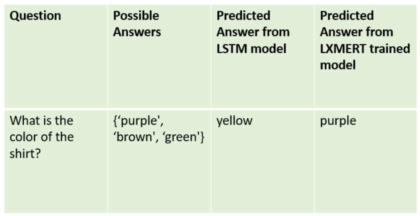
Predictions: Dog Lion Cat



Question: Are the people in the picture playing?
Ground Truth: Yes
Predictions: Yes



Question: What color is the hydrant?
Ground Truth: Red
Predictions: Blue Black Red
Here is a comparison of performance of our two models



| Question | Possible Answers | Predicted Answer from LSTM model | Predicted Answer from LXMERT trained model |
|---|---|---|---|
| What is the color of the shirt? | {'purple', 'brown', 'green'} | yellow | purple |

## 6 Conclusion and Future Work

The LSTM model we have implemented has a reasonable understanding of the questions and the answer images which can be seen improved by more advanced architectures like transformers. Since VQA is a relatively new research area, the benchmark datasets and accuracy metrics are constantly evolving. One of the future directions we would explore is the interpretability of the models, understanding the visual attention mechanism. This would help deduce how the models understand the images and in turn could also help in improving the model predictions.

## References

[1] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, Devi Parikh. VQA: Visual Question Answering *ICCV*, 2015. 1

[2] Mengye Ren, Ryan Kiros, Richard S. Zemel. Exploring models and data for image question answering *NIPS*, 2015. 1, 4

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2018. 2

[4] Jiasen Lu, Dhruv Batra, Devi Parikh, Stefan Lee. VilBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. *Advances in Neural Information Processing Systems*, 2019. 2

[5] Liunian Harold Li, Xiaodong Huang, Li Fei-Fei, Juan Carlos Niebles. VisualBERT: A Simple and Performant Baseline for Vision and Language. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019. 2

[6] Hao Tan and Mohit Bansal LXMERT: Learning Cross-Modality Encoder Representations from Transformers arXiv 2019 1

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998-6008). 2

[8] Mateusz Malinowski, Mario Fritz. Towards a Visual Turing Challenge *arXiv*, 2015. 1

[9] Licheng Yu, Eunbyung Park, Alexander C. Berg, and Tamara L.Berg. Visual Madlibs: Fill in the blank Image Generation and Question Answering *arXiv*, 2015. 2

[10] Shayan Hassantabar. Visual Question Answering: Datasets, Methods, Challenges and Opportunities *Princeton University*, 2018. 2

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollar. Microsoft COCO: Common Objects in Context *arXiv*, 2018. 2

[12] Mateusz Malinowski, Marcus Rohrbach, Mario Fritz. Ask Your Neurons: A Neural-based Approach to Answering Questions about Images *arXiv*, 2015. 2

[13] Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, Ram Nevatia. ABC-CNN: An Attention Based Convolutional Neural Network for Visual Question Answering *arXiv*, 2015. 2

[14] Hao Tan, Mohit Bansal. LXMERT: Learning Cross-Modality Encoder Representations from Transformers *arXiv*, 2019. 2

[15] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, Wei Xu. Are You Talking to a Machine? Dataset and Methods for Multilingual Image Question Answering *arXiv*, 2015. 2

[16] Sepp Hochreiter, Jürgen Schmidhuber. Long Short-Term Memory *Neural Computation*, 1997. 2

[17] Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan. Show and Tell: A Neural Image Caption Generator *arXiv*, 2015. 4

[18] Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan. DeViSE: A deep visual-semantic embedding model *NIPS*, 2013. 4

[19] Andrej Karpathy, Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions *CVPR*, 2015. 4

[20] Jeffrey Pennington, Richard Socher, Christopher Manning. GloVe: Global Vectors for Word Representation *EMNLP*, 2014. 4