Generative Visual Question Answering using Cross-Modal Visual-Linguistic Embeddings

John Chen
Department of Computer Science
University of Toronto
Toronto, ON
johnn.chen@mail.utoronto.ca

Yushi Guan
Department of Electrical and Computer Engineering
University of Toronto
Toronto, ON
yushi.guan@mail.utoronto.ca

Abstract

In this paper, we investigate the performance of several recent state of the art vocabulary-based Visual Question Answering models on the task of *answer generation*. *Answer generation* is a generalization of vocab-based QA, where the model must *generate* the answer token by token, and relaxes the strong assumptions made in the vocab-based formulation. We experiment with adding two different types of generative decoder heads to two state of the art vocab-based models, VL-BERT and LXMERT. In both cases, the decoder generator is not able to match the performance of the vocab-based original, although performance can remain competitive. Our findings indicate that simpler decoder heads can be more performant, demonstrating the importance of carefully crafting a training procedure or doing curriculum learning to jointly optimize the massive models. Finally, we confirm the intuition that answer generation is more challenging than answer selection and indicate that much progress remains to be made on the search for a general question answer system that is truly able to understand the question and express the answer fluently.

Keywords: visual question answering, self-attention, transformer, multi-modal fusion, joint modality pretraining, visual-linguistic embedding, BERT

1 Introduction

The Visual Question Answering (VQA) task lies at the intersection of visual and linguistic understanding. In VQA, given a question and image pair, the machine learning system needs to select an answer for the question based on information presented in the image [2]. The task is designed to test an artificial intelligence system's ability to reason using information from multiple modalities. There have been many models over the years that have improved the state of the art performance on VQA tasks [22][7]. End to end VQA models usually adapt models from vision and natural language researches and fuse latent information generated from the vision and language pipelines together, and select an answer based on the fused information.

The vocab selection-based formulation popularized by Antol et al. in 2015 is the dominant form of VQA: that of selecting the answer from a set of candidate answers [2]. However, there are some notable drawbacks to this problem formulation. In particular, it presumes that the correct answer is always known, and furthermore that it is presented as one of the candidate options. This artificially transforms the task of question answering to that of pattern matching. When humans think of question answering, they think of abstraction and reasoning; not just memorizing which answer keywords statistically co-occur with their corresponding questions.

In our work, we frame VQA as an answer generation task instead of an answer selection task. To do this, we tokenize the output answers so they become sequences of individual words. For the model architectures, we extend existing VQA solutions with different types of decoders to empower them

Preprint. Under review.

with answer generation capabilities. This formulation gives our model the capability to formulate unseen answers and exploitation of simple correlation between the input and the output should be more difficult.

Our solutions perform the VQA answer generation task by combining cross-modality embedding generation transformers with language sequence decoders. Inspired by the recent success of the Google BERT model, there have been a number of existing solutions that generate rich contextualized embeddings by combining both visual and linguistic information [6]. Some examples are LXMERT, VLBERT, VisualBERT and VilBERT [21, 16, 15, 19]. In our work, we focus on the pretrained LXMERT and VLBERT models.

We developed two types of language sequence decoders to effectively utilize the visual-linguistic embedding generated by LXMERT and VLBERT. We tested the performance of both a sequence-to-sequence type RNN decoder and a self-attention-based decoder similar to the one used in the transformers. A detailed architecture description is presented in section 3 and their performance results are presented in section 4.

2 Background and Related Work

2.1 Discriminative Visual Question Answering

We refer to the traditional vocabulary selection-based VQA as discriminative VQA in this paper. Since the introduction of the 2015 VQA dataset [2], this has been the standard approach to solve the problem and benchmark the performance. To tackle the problem, systems typically involve three components: a visual understanding component, a language comprehension component, and a component that fuses the information together and selects the final answer with a softmax layer. For example, Teney et al. used a faster R-CNN image feature extractor, a Gated Recurrent Unit (GRU) question embedding generator, and an attention-based multimodal fusion component [22]. Fukui et al. proposed a system that uses CNN for the image features, LSTM for the question, and a multimodal compact bilinear fusion component [7]. Despite improved performance over the years and differences in the details of the architectures, the overall objective and approach remained the same. While the original VQA v2 dataset paper argues for "free-form" answers, or multiple-word answers that are human generated, most current VQA systems simply coalesce these short answers into a pre-defined answer list.

The lack of creativity in the answers, and propensity for simple binary yes/no question has led to criticism regarding if these VQA tasks can be solved via simple data exploitation. For instance, [8] found that the Prior of the dataset (most common answer in the training set) is 25.98%, and a blind model that answers the question without access to the image can achieve 44.26% accuracy .

2.2 Generative Visual Question Answering

Many recent works have tried to address the issue of achieving high VQA performance via exploiting data correlations without higher-level reasoning. While there is no consensus on a standardized approach, these works tried to introduce some generative components into the whole process.

Hudson et al. introduced a new VQA dataset named GQA that minimizes the simple data correlation between the question and answer that a model can exploit [9]. The GQA dataset is created using an automatic question and answer generation engine, generating question and answer pairs that can only be solved via multi-hop reasoning. The automatically generated questions and answers also minimize the impact of data imbalance. **Wu et al. also explore generative question answering, where they use an RNN in order to generate answers [24]. This paper is essentially the spiritual successor of that paper.**

Lewis et al. proposed the Generative QA model that first generates the original question given the image features and the answer [14]. Then, Bayes' rule is used to reparametrize the distribution of answers given questions in terms of distribution of questions given answers. The authors argued that this process requires the model to perform multi-hop reasoning for both the answer and the question. This effectively converts the VQA task from a simple discriminative answer selection to a *question* generation task.

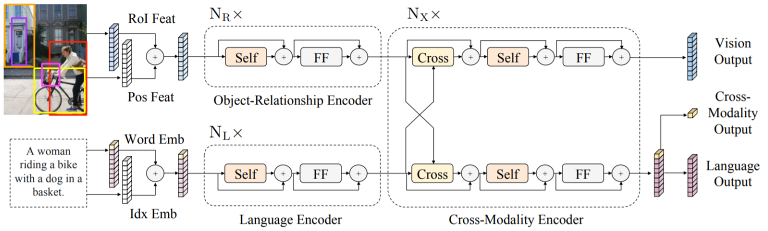


Figure 1: Architecture of LXMERT [21]

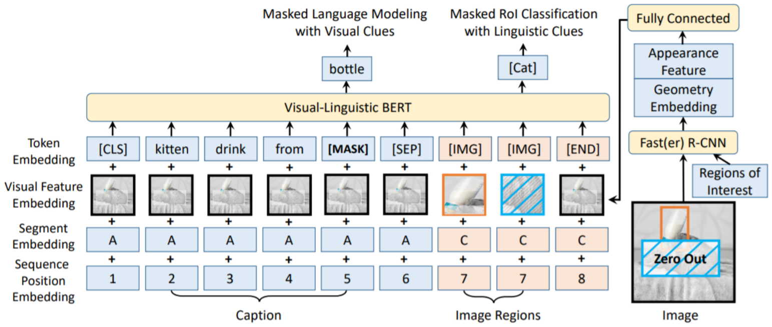


Figure 2: Architecture of VL-BERT [19]

2.3 Joint Visual-Linguistic Pretraining and Embeddings

The recent success of BERT in the natural language processing domain has given rise to a cornucopia of visual-linguistic BERT-based models including VilBERT, VL-Bert, VisualBERT, LXMERT and so forth. These models have established the state of the art on vision-and-language tasks, including vocab-based VQA. They capitalize on the recent proliferation of massive linked vision-and-language datasets, such as those for visual question answering, or image captioning. By pretraining on self-supervised tasks that leverage both linked modalities, these models learn useful joint visual-lingusitic embeddings.

2.3.1 LXMERT

Tan et al. proposed a pre-trained vision-and-language cross-modality framework [21]. The architecture is heavily inspired by Google's BERT pre-trained language understanding model based on self-attention layers. For background on the architecture and training of BERT, we refer readers to [6]. The LXMERT model architecture has three components: an object-relationship encoder (with a faster RCNN backbone), a language encoder and a cross-modality encoder [21, 1]. The language encoder is essentially the same as a BERT style self-attention based language encoder. The object-relationship encoder uses the Region of Interest (RoI) features extracted by faster RCNN backbones and processes them using a BERT-style encoder. The only difference is that the inputs are now image features instead of word embeddings. Lastly, a cross-modality encoder allows language and visual embeddings to attend to each other, fusing information together for downstream visual-language related tasks. A visual representation of the architecture is in Figure 1.

2.3.2 VL-BERT

VL-BERT is also inspired by the BERT architecture [19]. It gets rid of the canonical three-component solution for the VQA task (seen even in 2.3.1) and uses a two-component structure instead: a faster RCNN backbone and a fused visual-linguistic BERT-style encoder [1]. The faster RCNN extracts the RoI features just as LXMERT does. The visual-linguistic encoder fuses the input word embeddings of the question tokens and the extracted RoI features from the very first encoder layer, providing the two modalities more pathways to attend and combine information from each other. A visual representation of the architecture is in Figure 2.

3 Architecture Descriptions

We implemented several architectures to perform generative visual question answering.

3.1 LXMERT-Generative

We extend LXMERT in a straightforward fashion to incorporate a decoder. As shown in Fig 3a, we treat LXMERT as a multimodal encoder in a standard sequence-to-sequence framework, adding essentially just the decoder portion so that text may be generated. We selected the decoder architecture and hyperparameters according to the best performing combination from our baseline; more details can be found in the Experiments section. The decoder consists of a simple 3-layer GRU RNN with a 300-dimensional word embedding layer and final vocab layer of dimension $\|V\|$, where $\|V\|$ is the size of the vocabulary of the question and answer sequence pairs.

The input to the combined architecture, which we will refer to as LXG, is a natural language textual question and an image. These are both processed by the LXMERT-encoder (described in more detail in 2.3.1) which produces a final hidden representation summarizing the information contained in both modalities. This vector is then transformed by a bridging layer which converts the representation into a form amenable for our decoder, which then generates in the canonical sequence-to-sequence RNN fashion by computing $h_t = f(x_t, h_{t-1})$, where h_0 is the bridging layer representation. Finally, at each timestep, we decode one token by passing h_t through a vocabulary layer, and then taking a softmax, which is a discrete action to select an output token corresponding to a word. This output token is then reused in a canonical RNN fashion to generate the next token. Teacher-forcing is employed during the training procedure, which we describe in more detail in the Appendix.

We chose to keep our decoder simple in line with [13]. This allows us to more readily compare the effectiveness of LXMERT and other recent visual-linguistic models for generation compared to the previous SOTA analyzed by Lewis.

3.2 VLBERT with Meshed Decoder

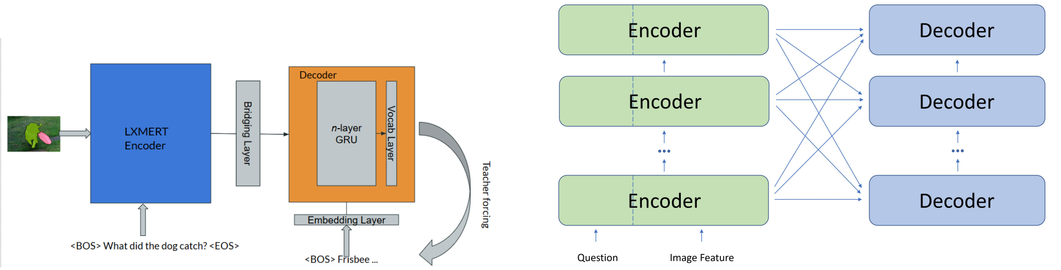
To bestow the VLBERT architecture with answer generation capabilities, we combine the VLBERT architecture with a transformer style decoder [23]. We refer to this architecture as VLM . The decoder uses embeddings extracted from the encoder layers, and outputs one word in the output answer at a time. At the first time step, the decoder uses only the encoder layer's output to generate the first token. At each of the following time steps, the decoder outputs a new token based on both the encoder layers' output and the previously decoded output. The decoding process do not stop until generation of the "<eos>" tag or the maximum sequence length. We used a total of 12 encoder layers so that we can utilize the pre-trained weights from VL-BERT. To decode the output, We use either 3 or 12 decoder layers. The 3 decoder layers version can be trained with a higher batch size to allow faster convergence but the 12 decoder layers version eventually reached better performance. Both the encoder and decoder has 12 heads and the hidden dimension is 768. The maximum output sequence length is set to 10.

We implemented our decoder such that each decoder layer has access to outputs from all the encoder layers. When there are only 3 decoder layers, they connect to only the last three encoder layer's output. This meshed style interconnection between the encoder layers and the decoder layers is inspired by Cornia et al.'s meshed transformer work, where they found that the meshed interconnect allows better information flow within the network and improved their model's performance on the image captioning task [5].

4 Experimental Setup and Results

4.1 Dataset and Task

We used the VQA v2 dataset for question answering [8]. The dataset contains 204,721 images from COCO. There are multiple questions associated with each image. Some questions such as "Is there



(a) We implement a generative decoder on top of LXMERT, which allows the answer to be *generated* as opposed to simply selected. At a high level, we can view LXMERT as simply acting as an encoder in a sequence-to-sequence encoder-decoder architecture.

(b) VLBERT with Meshed Decoder. The left side encoder is the same as the VLBERT encoder. The right side decoder is added to perform sequence generation. Each decoder is connected to all other encoders. In the 3 layer decoder version, they are connected to the last 3 encoder layers.

Figure 3: LXMERT and VLBERT with Decoder Model Architectures.

snow on the mountains?" require a simple "yes/no" answer. There are also questions that require a more generic answer, for example, "What is the child doing?". In total, there are 1,105,904 questions, with 10 answers given by different individuals for each question. The answers usually consist of one or a few words such as "yes", "yellow and red" and "file cabinet". There are also some longer answers such as "to keep pedestrians safe".

The input format to our architectures remain same as the LXMERT and VL-BERT approach. Our models take in the question and the extracted Region of Interest features from the original image. For the output, instead of treating the answer as one entity and let the model select it from a predefined list, we tokenize the output by splitting it into a sequences based on white spaces. The start and end of each sequence is tagged with a "<bos>" and an "<eos>" tag respectively. The decoder component of our models generate one token at a time; we do a greedy arg-max at each time step (beam search with size 1). This means that we simply select the word with the highest probability as our decoder output at time step t.

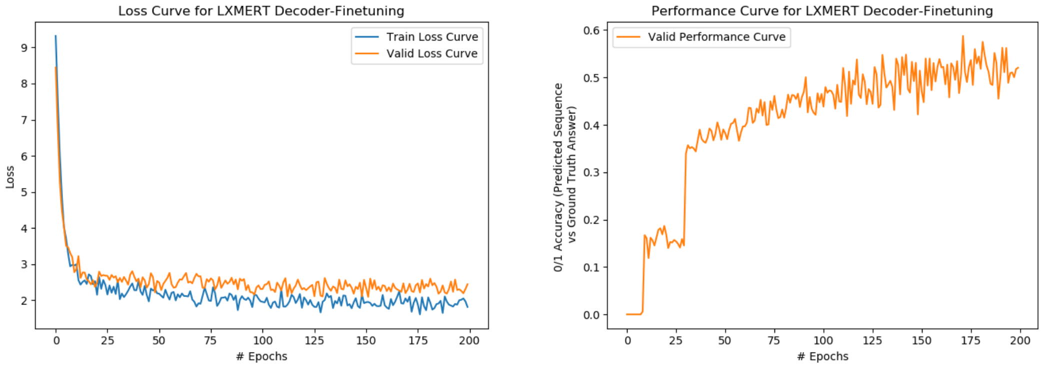
For evaluation, we report the strictest and most conservative measure of performance: 0/1 accuracy *on the entire sequence*. If the *entire output sequence* matches the original answer, we mark it as a correctly generated answer. This is in-line with [2, 13]; we do not consider using word-similarity measures like BLEU or ROUGE or word-embedding distance since they are not sensitive to precise distinctions like "left" vs "right".

4.2 LXG

Experiments are run on a single Titan Xp with 12 GB of memory. We test on a subset of the MSCOCO VQA dataset, using the "minival, traind, novalid" splits defined in [21]; furthermore, we use the "tiny" sub-setting (approx. 5000 examples only) due to speed and memory constraints. For all LXG "tiny" experiments, we use Adam [12] optimizer with learning rate set to 0.005, batch size of 40, and train for up to 200 epochs, using early stopping if necessary. We experiment with the batch size, doing word-level vs character-level encdoing and generation, size of the embeddings, RNN hidden dimension and other hyperparameters, as described in the Appendix. We also experiment with various types of transfer-learning: finetuning, where we train the decoder head from scratch, and optimize over all parameters and full-training, where we train both the encoder and decoder from scratch. Following best practices from [11, 21, 16], we also try doing stagewise training and curriculum learning. First, we train the decoder on a *blind* sequence-to-sequence mapping task between questions and answers, and once it is stable, combining the decoder with the encoder. More details are in the Appendix 8.2.

4.3 VLBERT with Meshed Decoder

Experiments are run on a single NVIDIA RTX 2070 GPU with 8 GB memory. We adapted the optimizer and learning schedule used in the original VLBERT paper. The adam optimizer is used,



(a) Loss curve on training vs validation set for LXG. Note that while the validation loss crosses above the training loss curve, both curves continue to decrease slowly, and furthermore the validation *performance* as shown on the right continues to improve.

(b) 0/1 accuracy is computed on the *sequence* – if any predicted word is wrong, then we count the entire answer as wrong. We draw attention to the rapid performance improvement that occurs around epoch 10 and epoch 35, indicating the all-or-nothing, discrete nature of our performance objective.

Figure 4: Loss and performance curves for LXG framework, using a pre-trained LXRT encoder and seq2seq decoder from scratch.

with an learning rate of 0.001 and a weight decay of 0.001. There is a warmup period which is commonly used in the training of transformers [19]. In our training, the warmup period involves 8000 steps. We use a batch size of 2 with the 12 decoder layers version and a batch size of 8 when 3 decoder layers are used. During training, the VLBERT encoders are being fine-tuned.

5 Results and Analysis

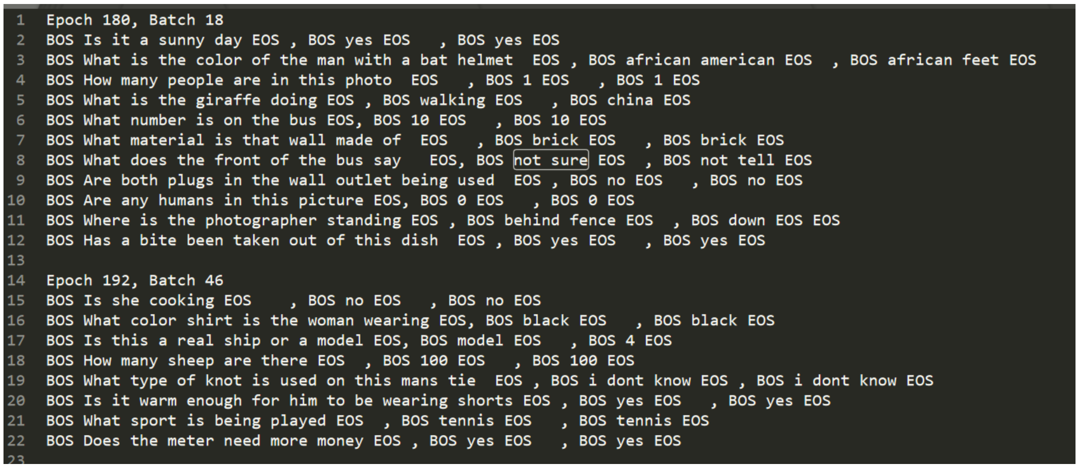
Table 1: VQA 0/1 Accuracy

Model	Augmentation	Crclum. Learning	Decoder Finetuned	Best Acc
Seq2Seq (text only)	-	-	39	39
LXG	Seq2Seq Decoder Only	48-51	52-56	65.2
	Seq2Seq Enc/Dec	49-52	48-51	-
VLBERT	3 Layer Decoder	-	41.3	41.3
	12 Layer Decoder	-	29.6	29.6
LXMERT	Baseline	-	-	**69.8**
VLBERT	Baseline	-	-	69.6

Table 2: Comparison of results of VQA accuracy performance for different architectures. Best Acc is reported wrt the *full* dataset: the best performing architecture is selected and then trained on the full training dataset, and evaluated on the *full* validation dataset. Details on the LXG Seq2Seq Enc/Dec variant can be found in the Appendix 7

5.1 LXG

Our best performing LXG model achieved a 0/1 accuracy of 65.2%, which is reasonably close to the performance of the vocab-based SOTA model. This LXG model uses only a finetuned decoder, and is trained for 10 epochs on the full dataset due to time limitations. We attribute our performance to several factors. Firstly, aiding the performance is the fact that a large portion of answers are 1-word answers: for instance, "<BOS> Yes <EOS> " or "<BOS> No <EOS> ". For these binary questions, answer generation essentially reduces to memorization/basic vocab-based VQA: instead of learning to select "No", the model learns to generate the one-word sentence "<BOS> No <EOS>".



```
1  Epoch 180, Batch 18
2  BOS Is it a sunny day EOS , BOS yes EOS   , BOS yes EOS
3  BOS What is the color of the man with a bat helmet  EOS , BOS african american EOS  , BOS african feet EOS
4  BOS How many people are in this photo  EOS   , BOS 1 EOS   , BOS 1 EOS
5  BOS What is the giraffe doing EOS , BOS walking EOS   , BOS china EOS
6  BOS What number is on the bus EOS, BOS 10 EOS   , BOS 10 EOS
7  BOS What material is that wall made of  EOS   , BOS brick EOS   , BOS brick EOS
8  BOS What does the front of the bus say   EOS, BOS not sure EOS  , BOS not tell EOS
9  BOS Are both plugs in the wall outlet being used  EOS , BOS no EOS   , BOS no EOS
10 BOS Are any humans in this picture EOS, BOS 0 EOS   , BOS 0 EOS
11 BOS Where is the photographer standing EOS , BOS behind fence EOS  , BOS down EOS EOS
12 BOS Has a bite been taken out of this dish  EOS , BOS yes EOS   , BOS yes EOS
13
14 Epoch 192, Batch 46
15 BOS Is she cooking EOS    , BOS no EOS   , BOS no EOS
16 BOS What color shirt is the woman wearing EOS, BOS black EOS   , BOS black EOS
17 BOS Is this a real ship or a model EOS, BOS model EOS   , BOS 4 EOS
18 BOS How many sheep are there EOS  , BOS 100 EOS   , BOS 100 EOS
19 BOS What type of knot is used on this mans tie  EOS , BOS i dont know EOS , BOS i dont know EOS
20 BOS Is it warm enough for him to be wearing shorts EOS , BOS yes EOS   , BOS yes EOS
21 BOS What sport is being played EOS  , BOS tennis EOS   , BOS tennis EOS
22 BOS Does the meter need more money EOS , BOS yes EOS   , BOS yes EOS
23
```

Figure 5: Generated answers for LXG when training on the full dataset. Data is delimited by commas and BOS EOS tags; the first split is the question, the next split is the ground truth answer, and the final split is the actual answer generated by the model, using beam-search with width 1 (greedy argmax). As we can see, the model is able to correctly *generate* multi-word answers (line 19), although not always (line 3, line 8).

Naturally, the other side of the coin is that some answers are lengthier and comprise a few words. In this setting, the task necessarily becomes harder: both because there are *multiple* words to predict correctly, and because of the recurrent and *discrete nature* of text generation. Because each token is computed as a function of the previous token, and because we use beam-search with size 1, if a mistake is made early on, it can be difficult for the decoder to decode the rest of the string. As shown in Fig 5, multiple types of errors can be made in the generation of multiword answers. In lines 3 and 8, we see examples of substitution errors ("african **feet**" instead of "african **american**", and "not **sure**" instead of "not **tell**") and in line 11 we see an example of a compound substition and deletion error ("behind fence" vs "*down*").

Finally, we saw a big improvement in accuracy when training on the full dataset, as shown for the seq2seq best accuracy in table 2. Empirically, we also found that increasing the batch size was also extremely beneficial to learning as well, in line with [23]. This suggests the importance of engineering and training procedures in final performance.

5.2 VLBERT with Meshed Decoder

The 12 layer decoder and 3 layer decoder version of the proposed VLBERT with Meshed Decoder reached an accuracy of 41.3% and 29.6% respectively. Despite using a more complex Transformer architecture for the decoder, the performance is lower compared to LXG. We believe the low performance can mainly be attributed due to difficulties in training the network and additional overhead from using a much more computationally intensive decoder. While LXG uses a batch size of 40, the large and small transformer-style decoders use respective batch sizes of just 2 and 8. Since we are decoding across multiple timesteps, it is especially important that we keep a large batch size to maintain stability.

As the visual-linguistic papers demonstrate, large transformers and BERT models also require more careful and bespoke training procedures [15, 16, 21], including requiring precise learning rate schedule tuning to perform optimally [23, 6]. As demonstrated in Cornia et al.'s transformer for image captioning, smaller transformers still have the expressivity to capture information in the visual and linguistic elements [5]. As we show in another experiment in the Appendix 7, we also experiment augmenting LXG with a *second* encoder and similarly find that a larger architecture is not always better. Evaluation of the performance of a 3 layer encoder version would require a new pre-train of the VL-BERT encoder which takes substantial amount of time and computation resources we do not have but it is a plausible future work.

5.3 Qualitative Evaluation of Results

Fig 5 displays some generated outputs of LXG, from different periods of the training. Examining these generated results, we find that we are able to explain the rapid performance improvements at certain epochs observed in 4; the model learns to correctly “memorize” the one-word answers such as “<BOS >Yes <EOS >”. Note that in this situation, there is a one-to-one correspondence with the vocab based approach and the generative approach; although the added bonus is that the model also learns the semantics of the special <BOS >and <EOS >symbols. As discussed in 5.1, LXG benefits immensely from the multitude of simple one-word answers, but is also able to generate nice multi-word answers occasionally.

Similarly, for the generated outputs of VLBERT with Meshed Decoder, we observe that it can sometimes capture very complicated answers such as "<bos> hot dog onions and potatoes <eos>", but it also makes a lot of mistakes including generating "yes/no" answers frequently when the ground truth are actually other answers. More results are included in the Appendix 8.

6 Limitations and Future Extensions

6.1 Adopting NLP best-practices and improving engineering

Our work is very exploratory and can be improved in many ways. First, we do not employ the full toolbox of NLP training techniques to unlock maximum performance. For the decoding process in particular, we do not use a beam search (other than greedy argmax) to find the best outputs, and do not perform reinforcement learning fine-tuning, both of which are common techniques that often empirically improve performance. Applying these best practices from the NLP literature, along with engineering improvements such as increasing batch size, would be an immediate future improvement. While LXG currently performs better than VLM, we would expect VLM without memory and computation limitations to work better in the future, due to the meshed attention mechanism. In particular, some form of attention is a critical component [3, 17, 23], especially for longer, multi-hop reasoning for questions in newer datasets like [10].

6.2 Towards semantic VQA evaluation

A more fundamental limitation of VQA and our models in general is dealing with multiple correct answers. Currently in our approach, for every image and question pair, we assume there is a *single* ground-truth answer, even though multiple answers are provided in the VQA v2 dataset itself. This impacts the effectiveness of our model, since potentially, similar answers may also be semantically similar in word-embedding space for instance and our model may be able to leverage its internal implicit language model to discover this. However, as [2] mention in their original VQA paper, relying too much on sequence-level similarity has its own drawbacks: an answer like "take the turn on the left" is conceptually the opposite of "take the turn on the right", even though they share many words of overlap. Creation of a textual "perceptual" loss, or semantic loss for VQA remains an open problem, and will be critical to advancing the state of the art in the future.

7 Conclusion

In this paper, we explore the topic of *generative* visual question answering, updated for current state of the art VQA methods which use joint visual-linguistic embeddings. We implement two generative decoder head architectures on two SOTA vocab-based VQA models, LXMERT and VL-BERT, and show comparable performance to the vocab-based analogues. We empirically determine good settings for the decoder hyperparameters and architecture, and show that additional model capacity may indeed hurt performance due to the scale and complexity of the pre-training task. We provide qualitative analysis of errors and provide guidance for future research directions. By formulating question answering as sequence-to-sequence learning, instead of vocab-based pattern matching, we dispose with the unnatural and artificial constraint that the question being asked must have an answer *that the model has seen before*. Our work opens the door for more research into creative question answering and reasoning for intelligent agents.

References

[1] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering, 2017.

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

[5] M. Cornia, M. Stefanini, L. Baraldi, and R. Cucchiara. Meshed-memory transformer for image captioning, 2019.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding, 2016.

[8] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering, 2016.

[9] D. Hudson and C. Manning. Gqa: a new dataset for compositional question answering over real-world images, 02 2019.

[10] D. A. Hudson and C. D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6700–6709, 2019.

[11] J. Johnson, B. Hariharan, L. Van Der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2989–2998, 2017.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] M. Lewis and A. Fan. Generative question answering: Learning to answer the whole question. 2018.

[14] M. Lewis and A. Fan. Generative question answering: Learning to answer the whole question. In *ICLR*, 2019.

[15] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang. Visualbert: A simple and performant baseline for vision and language, 2019.

[16] J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23, 2019.

[17] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation.

[18] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

[19] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

[20] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[21] H. Tan and M. Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

[22] D. Teney, P. Anderson, X. He, and A. van den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge, 2017.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[24] Q. Wu, P. Wang, C. Shen, A. Dick, and A. Van Den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4622–4630, 2016.

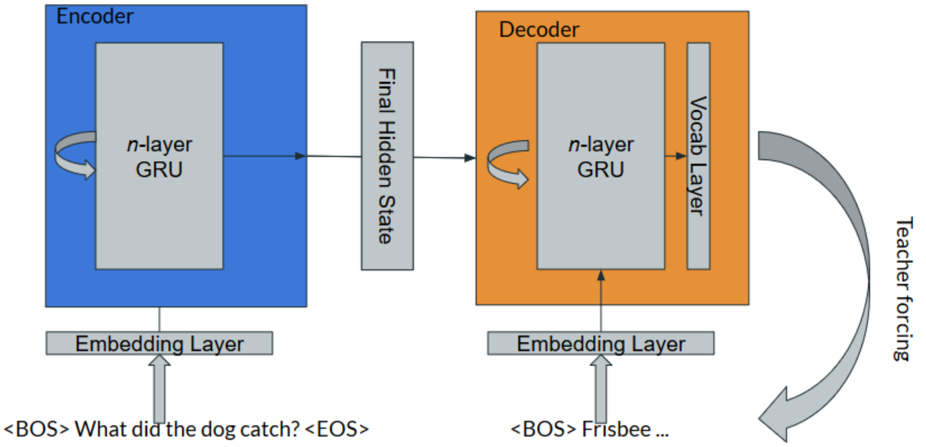


Figure 6: General encoder-decoder sequence-to-sequence framework. This model maps textual sequences (questions) to other textual sequences (answers), even though it is *blind*; it has no access to visual modality.

8 Appendix

8.1 Extended background

8.1.1 Transformers and other Self-attention architectures

Self-attention models have recently demonstrated superior performance in language understanding as seen in the Transformers (machine translation) and in the BERT (contextualized language representation). It has then been applied to visual problems and has reached performance similar to CNN-based architectures in some tasks [18]. In self-attention models, no convolution or recurrent layers are being used, instead, it calculates the value representation of each input, as well as the attention scores which indicate how much "attention" each input should pay to other elements in the input. The output of a self-attention mechanism is the weighted sum of the value representation based on the attention scores.

8.2 Curriculum Learning the LXG architecture

For our LXG architecture, we performed several curriculum learning procedures to find a good performing final model.

8.2.1 seq2seq baseline

We begin by implementing a baseline sequence-to-sequence (seq2seq) model, which we use as a building block later to extend LXMERT. Introduced in 2014 by Sutseker et al. [20], seq2seq is a conceptually straightforward framework for formulating sequence-to-sequence mapping problems. At a high level, the seq2seq framework consists of an encoder block and a decoder block. The encoder processes the input sequence, timestep by timestep, generating a final hidden representation, which is then fed to the decoder. Starting from the final hidden state of the encoder, the decoder then sequentially and recurrently generates the output sequence.

For our instantiation of the seq2seq baseline, we use a GRU [4] encoder RNN and a GRU decoder RNN, each of which have three hidden layers, and hidden dimension of 300. Architecture and other hyperparameter settings were determined via experimentation. In both encoder and decoder, we encode the text into a 300 dimensional word-embedding; in our decoder network, we furthermore have a vocab layer which maps the hidden representation back into a token for decoding. We ran a wide array of experiments on our baseline, testing word level vs character level performance,

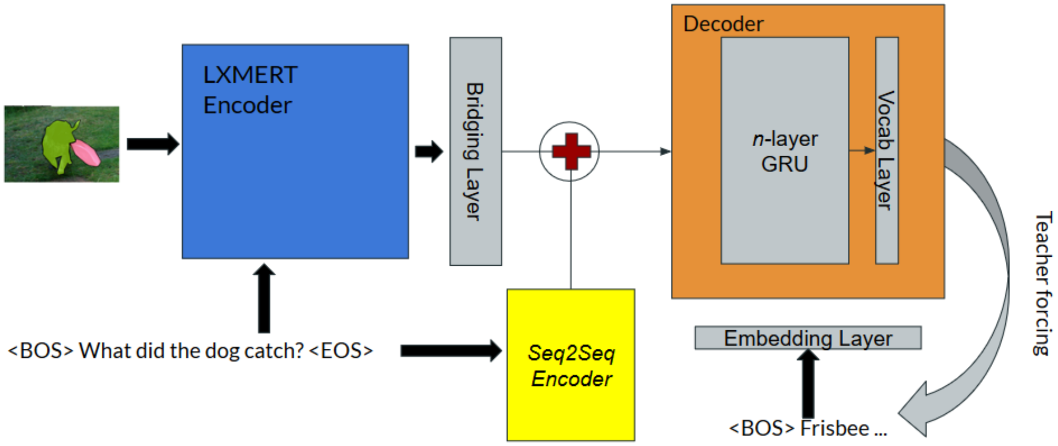


Figure 7: Double encoder LXG architecture. The red cross indicates an element wise addition; care must be taken to ensure that the bridging layer representation and Seq2Seq encoder final hidden state representation have matching dimensions.

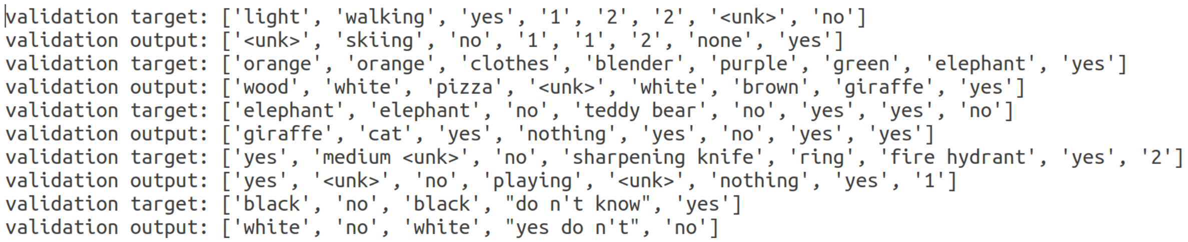
different batch sizes, dimensions and sizes of our networks, bidirectionality of the RNN, different RNN architectures (GRU, LSTM, vanilla RNN) and teacher forcing hyperparameter.

8.3 Redundancy of LXMERT embeddings

To test the idea that additional model capacity may not be beneficial, we also experiment with an additional architecture, where we use *two* encoders. This model performs worse. This is surprising considering that additional model complexity should make the performance no worse – in the worst case, the model should learn to simply zero out the connection if it truly does not add to performance. However, due to the stochastic and non-convex training procedure, we likely see this behaviour.

Our negative results with VLM, further support this hypothesis. While in theory additional model capacity should help, in practice, the additional overhead of the decoder computational cost and optimization of so many parameters for the joint task is too tricky to allow for this.

8.4 VLM results



```
validation target: ['light', 'walking', 'yes', '1', '2', '2', '<unk>', 'no']
validation output: ['<unk>', 'skiing', 'no', '1', '1', '2', 'none', 'yes']
validation target: ['orange', 'orange', 'clothes', 'blender', 'purple', 'green', 'elephant', 'yes']
validation output: ['wood', 'white', 'pizza', '<unk>', 'white', 'brown', 'giraffe', 'yes']
validation target: ['elephant', 'elephant', 'no', 'teddy bear', 'no', 'yes', 'yes', 'no']
validation output: ['giraffe', 'cat', 'yes', 'nothing', 'yes', 'no', 'yes', 'yes']
validation target: ['yes', 'medium <unk>', 'no', 'sharpening knife', 'ring', 'fire hydrant', 'yes', '2']
validation output: ['yes', '<unk>', 'no', 'playing', '<unk>', 'nothing', 'yes', '1']
validation target: ['black', 'no', 'black', "do n't know", 'yes']
validation output: ['white', 'no', 'white', "yes do n't", 'no']
```

Figure 8: Some generated answers from VLM architecture