Towards Open-Ended VQA Models Using Transformers

Alberto Mario Bellini[1,2] , **Natalie Parde**[2] , **Matteo Matteucci**[1] and **Mark James Carman**[1]
[1]Politecnico di Milano
[2]University of Illinois at Chicago
albertomario.bellini@mail.polimi.it, parde@uic.edu
matteo.matteucci@polimi.it, mark.carman@polimi.it

Abstract

Visual Question Answering (VQA) is an open area of research at the intersection of natural language processing and computer vision, with many exciting practical applications. However, most prior work on this problem shares a standard limitation: the number of possible answers is restricted to a limited set of candidates, constraining the resulting models' power and flexibility. We address this shortcoming by introducing a new architecture for VQA that employs a state-of-the-art language model, the Transformer, to generate open-ended answers. This novel approach allows the system to generate unconstrained text rather than selecting from discrete categories, representing a substantial contribution towards developing more humanlike VQA systems. We show that our architecture compares well with existing VQA models, setting an exciting new benchmark for future research.

1 Introduction

Visual Question Answering (VQA) is an open, multidisciplinary research problem that combines computer vision, natural language processing, and deep learning to interpret and respond to open-domain questions about images. This task is challenging because it requires both (a) an ability to understand what has been asked, and (b) an ability to reason on the associated image to seek relevant information. If both of these subtasks are performed adequately, a sound VQA system should be capable of generating meaningful answers to the questions in natural language, preserving semantic and syntactic correctness.

Despite the difficulty of these interrelated subtasks, current VQA architectures achieve strong results and can generalize reasonably well on different question-image pairs [Antol *et al.*, 2015; Andreas *et al.*, 2016; Lu *et al.*, 2016; Hu *et al.*, 2017; Tan and Bansal, 2019]. Still, they share a standard limitation: the generated answers tend to be short and concise (e.g., "yes," "no," "blue"), with little to no context. This arises at least partially from how the VQA problem is commonly framed—language generation is typically treated as a classification task, rather than the open-ended, unconstrained problem more reflective of how it is approached by humans. Because of this framing, answers are forced to lie within a limited, predefined set of candidates. This naturally creates a dichotomy between the human and machine scenarios—while humans may enrich their answers, providing longer sentences with punctuation and arguments, VQA systems tend to rely on outputting single words as a performance-preserving mechanism.

In this work we tackle VQA using a different approach: instead of distributing probabilities over a predefined set of possible candidate answers, we employ a Generative Pre-Trained Transformer-2 (GPT-2) model [Radford *et al.*, 2019] to generate answers one token at a time. GPT-2 is a powerful, Transformer-based language model that relies entirely on attention mechanisms to condition the generation of tokens on the given context and the tokens generated at previous time steps. Our contributions are as follows:

1. We leverage the Transformer architecture in a VQA system to develop a model capable of exploiting its linguistic knowledge to generate more natural and open-ended answers.
2. We empirically demonstrate the feasibility of combining GPT-2 with common visual feature extractors (e.g., convolutional neural networks), to condition the generation of each token not only on the textual modality but on visual input as well.
3. We do so by introducing a variation of the Transformer architecture such that feature maps output from the image encoder are taken into consideration when generating the next token.
4. We show that our novel architecture is competitive with existing VQA models, providing evidence that unconstrained, generative models are a viable approach for VQA and establishing a benchmark to stimulate further research in this area.

We describe our methods and evaluation procedures in more detail in the following sections.

2 Related Work

Prior work in VQA has primarily shared a common structure [Antol *et al.*, 2015; Andreas *et al.*, 2016; Lu *et al.*, 2016; Hu *et al.*, 2017; Tan and Bansal, 2019]. First, encoders (e.g., long short-term memory (LSTM) or convolutional neural network (CNN) models) extract features from the visual and textual inputs. Then, the two modalities are joined in a common subspace to form a higher-level representation. Finally, a classifier distributes probabilities over a vocabulary of possible answers. A host of smaller variations to this common structure exist across architectures; we describe the highest-performing recent models in more detail.

MCB. Multimodal Compact Bilinear Pooling (MCB) [Fukui *et al.*, 2016] is a robust architecture that exploits an outer product to combine the two modalities, to maximize the multiplicative interaction between their two vector representations. Fukui *et al.* [2016] devised this mechanism under the hypothesis that existing methods for combining the two modalities were insufficient for projecting them in the right space. In MCB, the vector representations for text and visual inputs are computed, respectively, by a two-layer LSTM and a Residual Network [He *et al.*, 2016]. Once the multimodal representations are encoded, they are pooled together. Traditional bilinear pooling results in high-dimensional vectors; however, MCB efficiently compresses the output for every modality by first projecting the image and question into a higher-dimensional space using the Count Sketch method [Charikar *et al.*, 2002], and then convolving the two representations by taking the element-wise product in the Fast Fourier Transform space [Gao *et al.*, 2016]. Given the attention output computed by MCB and the outputs of the LSTM, the final answer is generated using a classifier. MCB achieves a 62% accuracy on the VQA dataset.

BUTD. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering [Anderson *et al.*, 2018] is a recent architecture that currently achieves an accuracy of 70.3% on the VQA dataset. It combines bottom-up and top-down attention mechanisms that enable attention to be calculated both at the level of objects and other salient image regions. The bottom-up attention mechanism accounts for determining interesting regions in the input image, whereas the top-down mechanism determines how to weight each region. The final classification model distributes probabilities over a fixed set of candidate answers.

LXMERT. Tan and Bansal's [2019] architecture is the current state-of-the-art on the VQA dataset, achieving a 72.5% accuracy. Like our approach, LXMERT incorporates a Transformer architecture, although it does so for a different purpose. The large-scale model consists of three encoders: an object relationship encoder, a language encoder, and a cross-modality encoder. To train the model to connect vision and language, Tan and Bansal pre-train the model with large amounts of image-and-sentence pairs. However, the model does not generate open-ended answers, with the authors instead exploiting Transformers as more powerful encoders. Ultimately, the cross-modal intermediate output is fed to a classifier, which still distributes probabilities over a predefined vocabulary of answers.

3 Architecture

In this work, we introduce a generative VQA model (VGGPT-2) that employs attention mechanisms at multiple levels and is capable of using the information contained in the question to focus on specific portions of the image. This information is in turn used to mask out irrelevant portions of the question to facilitate the generation of accurate answers. Four components work together to process both the visual and textual modalities to form natural, unconstrained answers. We describe these components in the subsections below, and provide a high-level overview of the architecture in Figure 1.

3.1 Image Encoder

The first component, an image encoder, consists of a pre-trained VGGNet-11[1] model [Simonyan and Zisserman, 2014]. We employ this model to extract visual features from the images associated with the input questions. Since, in our context, we do not need VGGNet to act as a classifier, but rather as a feature extractor, we follow common practice and drop the last three fully-connected layers, retaining only the output of the last average pooling layer. This way, given an image, the encoder produces a set of maps that contain high-level extracted features.

Specifically, letting c_i be three different channels (red, blue, and green) of width $w_i = 224$ and height $h_i = 224$, the visual input consists of an image $\mathbf{I} \in \mathbb{R}^{c_i \times w_i \times h_i}$. The output of the image encoder then consists of a set of maps $\mathbf{M} \in \mathbb{R}^{c_o \times w_o \times h_o}$, where $c_o = 512$ different channels of width $w_o = 7$ and height $h_o = 7$. $\mathbf{M}$ then serves as one of the two inputs to our attention mechanism.

3.2 Language Model

We use OpenAI's GPT-2 (small, 117M) Transformer [Radford *et al.*, 2019] as our language model, refining their pre-trained architecture in order to integrate it into our system. GPT-2 consists of two core blocks: (1) the Transformer, which is responsible for computing the multi-head attention discussed previously; and (2) the final classifier which, for each token processed by the Transformer, produces a distribution of probabilities over the vocabulary of words.[2]

Since our goal is for the language model to interact with the image encoder, we split apart GPT-2's two blocks and use them separately in our model. Given a sequence of n input tokens $\mathbf{S} \in \mathbb{R}^{n \times 1}$, the output of the transformer $\mathbf{T} \in \mathbb{R}^{n \times hidden}$ consists of a sequence of n hidden vectors of size $hidden = 768$. We use $\mathbf{T}$ as the second input to our attention mechanism.

3.3 Attention

The attention component is critical to our architecture, and is responsible for combining features from the two different sub-spaces into a single representation. We employ a co-attention mechanism, using the question to drive the focus on the image and vice versa. To formally define this component, we recall its inputs:

[1]We do not experiment with deeper vision architectures in this work since our primary focus is on the language modeling aspect.

[2]Recall that this is a language model that exploits attention, which means that the probability $p(w_i|t_i, t_{i-1}, ...t_{i-k})$ of generating the word w_i at time step i is conditioned on the input tokens $t_i, t_{i-1}, ...t_{i-k}$ up to time step i.

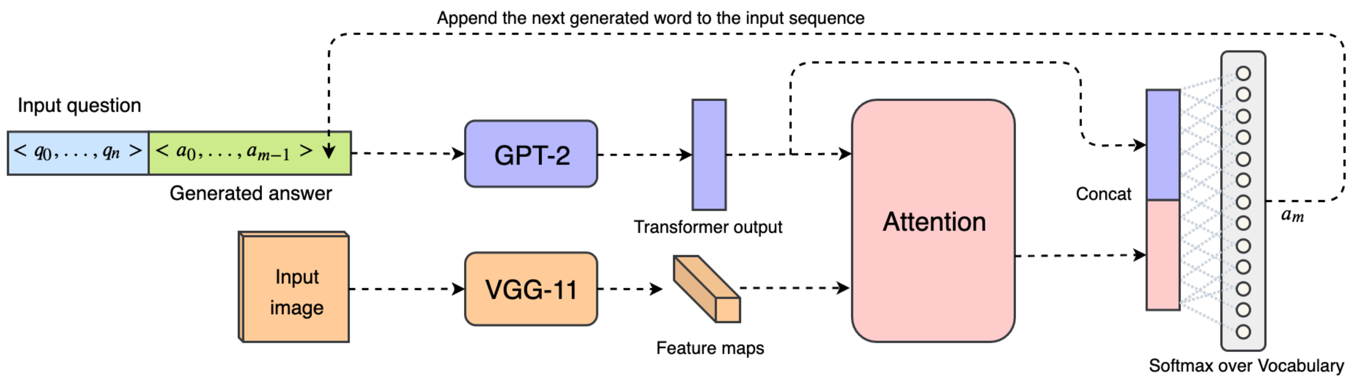


Figure 1: VGGPT-2 architecture overview, highlighting the answer generation process beginning from a question/image pair.

- **Image Encoder Output**: a vector $\mathbf{M} \in \mathbb{R}^{512\times7\times7}$
- **Language Model Output**: a vector $\mathbf{T} \in \mathbb{R}^{n\times768}$

Our model first flattens **M** into a more convenient representation $\mathbf{M_{flat}} \in \mathbb{R}^{512\times49}$. Then, both $\mathbf{M_{flat}}$ and **T** are fed to two separate linear layers ($\mathbf{Lin_M}$ and $\mathbf{Lin_T}$) that bring the two vectors into a common subspace of size $AttDim = 512$:

- $\mathbf{Lin_M}$: brings $\mathbf{M_{flat}} \in \mathbb{R}^{512\times49}$ into $\mathbf{M_{att}} \in \mathbb{R}^{AttDim\times49}$
- $\mathbf{Lin_T}$: brings $\mathbf{T} \in \mathbb{R}^{n\times768}$ into $\mathbf{T_{att}} \in \mathbb{R}^{n\times AttDim}$

At this point, for every element $\mathbf{T_{att}}^{\mathbf{i}} \in \mathbb{R}^{AttDim}$ in $\mathbf{T_{att}}$, with $i \in \mathbb{N} \wedge i \in [0, n)$, we repeat the following set of operations:

1. We sum $\mathbf{T_{att}}^{\mathbf{i}}$ to each element $\mathbf{M_{att}}^{\mathbf{k}} \in \mathbb{R}^{AttDim}$ of $\mathbf{M_{att}}$, with $k \in \mathbb{N} \wedge k \in [0, 49)$, obtaining the first multi-modal attention representation $\mathbf{A_1}^{\mathbf{i}} \in \mathbb{R}^{AttDim\times49}$ for the i^{th} token in the input sequence.
2. We rectify $\mathbf{A_1}^{\mathbf{i}}$ using a *ReLU* unit and obtain an intermediate attention representation $\mathbf{A_2}^{\mathbf{i}} \in \mathbb{R}^{AttDim\times49}$.
3. Using a linear layer ($\mathbf{Lin_{soft}}$) we compress $AttDim$ down to 1. In other words, $\mathbf{Lin_{soft}}$ brings $\mathbf{A_2}^{\mathbf{i}} \in \mathbb{R}^{AttDim\times49}$ into $\mathbf{A_3}^{\mathbf{i}} \in \mathbb{R}^{1\times49}$.
4. To obtain a softmap that highlights relevant spots of the image given a specific word at position i, we pass $\mathbf{A_3}^{\mathbf{i}}$ through a $Softmax$ function and obtain $\mathbf{A_4}^{\mathbf{i}} \in \mathbb{R}^{1\times49}$, where $\mathbf{A_4}^{\mathbf{i,h}} \in (0, 1)$ with $h \in \mathbb{N} \wedge h \in [0, 49)$. At this point every pixel in $\mathbf{A_4}^{\mathbf{i}}$ has an associated value between 0 and 1, which indicates how important that pixel is to answer the question.
5. We perform a **point-wise multiplication** between every element $\mathbf{M_{flat}}^{\mathbf{k}} \in \mathbb{R}^{1\times49}$ in the original flattened maps $\mathbf{M_{flat}}$ with $\mathbf{A_4}^{\mathbf{i}}$, obtaining a new vector $\mathbf{A_5}^{\mathbf{i}} \in \mathbb{R}^{512\times49}$. In this step we perform the **Question-to-Image attention**, since we exploit the softmaps computed at the previous step to mask out irrelevant portions of the image.
6. We sum, for each channel in $\mathbf{A_5}^{\mathbf{i}}$, all the masked pixels together, reducing the dimension from 49 to 1. In other words, we go from $\mathbf{A_5}^{\mathbf{i}} \in \mathbb{R}^{512\times49}$ to $\mathbf{A_6}^{\mathbf{i}} \in \mathbb{R}^{512\times1}$ by summing all the elements together on the pixel axis.
7. We bring $\mathbf{A_6}^{\mathbf{i}}$ back to the Transformer space through a final linear layer ($\mathbf{Lin_{out}}$), which expands $\mathbf{A_6}^{\mathbf{i}} \in \mathbb{R}^{512\times1}$ to $\mathbf{A_7}^{\mathbf{i}} \in \mathbb{R}^{768}$.
8. We compute the **Image-to-Question attention** through a final **point-wise multiplication** between $\mathbf{A_7}^{\mathbf{i}}$ and every element $\mathbf{T}^{\mathbf{i}} \in \mathbb{R}^{768}$ of **T**, which is the original output of the Transformer before this layer. The final output of the attention mechanism for every embedded token $\mathbf{T}^{\mathbf{i}}$ and the maps **M** coming from the image encoder is the vector $\mathbf{A_{out}}^{\mathbf{i}} \in \mathbb{R}^{768}$.

We described how the attention layer works considering one single embedded token $\mathbf{T}^{\mathbf{i}}$ at a time. However, it is easy to scale it up to a sequence of n elements. To do so, we just include an additional dimension for the vectors in the attention layer that accounts for the token under consideration. The output $\mathbf{A_{out}}$ of the attention mechanism is then combined back with the output of the language model to generate the answer.

3.4 Answer Generator

We generate the final answer by once again exploiting the language model. This last component diverges from traditional VQA systems—while most existing work has fed the output of the attention mechanism to a classifier, thereby distributing probabilities over a predefined set of answers, we frame this last step differently.

We condition the generation of words using the output of the attention layer. Instead of feeding the output **T** of the Transformer directly to its **head**, we first perform a key operation: we take the attention output $\mathbf{A_{out}}$ and concatenate it with the Transformer output **T** into a new vector $\mathbf{O} \in \mathbb{R}^{n\times1536}$. Then, with the help of a new linear layer ($\mathbf{Lin_{Classifier}}$), we distribute probabilities over the original vocabulary of words from the pre-trained GPT-2 model. In order not to lose the valuable information contained in the weights of the pre-trained model **C** (or **head**), we do the following:

1. We initialize the first 768×50254 weights of $\mathbf{Lin_{Classifier}}$ with the ones present in the original **head** of GPT-2.
2. We initialize the last 768×50254 weights, accounting for the attention layer output, to 0.

This allows the model to start training from a state in which it is already able to generate sequences of words correctly, since all weights relevant to the attention mechanism are initialized to 0, and the only ones that influence the output are coming from **T**. This last linear layer consists of roughly 77.1M parameters (1536×50254), so it is imperative to perform such an initialization to train the system in a reasonable amount of time.

Afterward, we feed **O**, also known as the **Logits** vector, to a *Softmax* function that distributes probabilities over our vocabulary. To generate the output word w^i corresponding to the input token at position i, we pool the most probable word in a greedy fashion.[3] Importantly, instead of using a compact representation of the input question and the image to distribute probabilities over a fixed set of answers, we dynamically generate each word, conditioning it on the previously-generated words. At evaluation time, this allows the system to generate arbitrarily long answers in the following fashion:

1. Given a question of n words and an image, we first let the system generate the initial word in the answer, conditioned only on the question and the image.
2. After we have generated the first word, we append it to the original question, which becomes a sequence of $n+1$ words. This time, the second word in the answer gets generated conditioned on the new sequence.
3. The procedure goes on until the system outputs the <***eos***> token or a maximum length threshold is reached.

4 Dataset

Following other recent work on VQA, we train and evaluate our model using the Visual Question Answering V2 (VQAv2) dataset [Goyal *et al.*, 2017], a large collection of real-world question-image pairs. Although the corpus is primarily meant for standard classifier-driven VQA models, its use facilitates direct comparison with existing models; such a comparison is key to demonstrating the effectiveness of open-ended, generative VQA models such as that proposed here.

Importantly, the dataset is based on real-word images and not on synthetic ones, allowing for better generalization capabilities. It consists of more than 200K images, 1M questions and 10M ground truth answers, manually annotated. There are at least 3 questions (5.4 on average) per image, and each question/image pair has 10 different annotated answers. To disincentivize answers based purely on linguistic bias, some questions have complementary image pairs. Additionally, answers were balanced for different question types in order not to have a predominant value.

4.1 Preprocessing

We apply a variety of preprocessing steps to the text and images in order to most productively make use of the corpus. We summarize the structure of the preprocessed dataset in Table 1, and elaborate further on our preprocessing steps below.

[3]We also experimented with beam search, finding that with beam sizes greater than one, the model generated shorter answers more characteristic of those produced by traditional classification-based VQA systems.

	Train	Test
Split size	10^6	10^5
Sequence length	24	22
Question + Answer	yes	no
Min. question length	5	6
Max. question length	21	22
Avg. question length	8.45	9.48
Min. answer length	3	-
Max. answer length	18	-
Avg. answer length	3.89	-
Avg. <***pad***> count	11.65	-
# of unique images	71450	32822

Table 1: Structure of the processed dataset

Grayscale removal. We discarded all samples associated with non-RGB images, retaining only colored images.

Answer length. Since we were interested in building an architecture capable of generating open-ended answers, we prioritized longer samples and gave less importance to those with concise answers.

Answer variability. We randomly selected up to 4 annotations for each question, ensuring that at most 4 identical question/image pairs were fed to the model, with 4 different answers.

Data formatting. Given a question/answer pair for a given image, we concatenated the pair to form a sequence of words. This sequence was then fed to the model at training time. For the test data, we dropped all answers and kept only the question/image pairs. At evaluation time we let the model generate the full answer in a greedy fashion, by pooling the most probable word at each time step. All textual inputs were byte-pair encoded [Shibata *et al.*, 1999] using the pre-trained GPT-2 vocabulary.

Custom tokens. We introduce four new tokens: <***bos***>, <***eos***>, <***sep***>, and <***pad***> to indicate the beginning and end of the sequence, the separation between question and answer, and finally the padding token. In the training set, we make use of all four tokens while encoding the input sequence. In the test set, we only use <***bos***> and <***sep***> since no answer is present. Since we introduce these new tokens, we set the language model's weights to be trainable in order to update the embedding matrix and the attention layers.

5 Training

To train our architecture we employ a technique called Teacher Forcing [Lamb *et al.*, 2016]: given a question with its associated answer, we combine the two to form a sequence of tokens that are fed to the model. Exploiting a *CrossEntropy* loss function, we train our system to predict the input sequence shifted by 1 time step, effectively telling the model to condition the probability of generating a new token on the previously seen context. The loss function ignores the <***pad***> tokens and focuses only on portions of the sequence that contain meaningful information. We block weight updates in our image encoder, using it solely as a visual feature extractor. However, we set our language model's weights to be trainable. This is necessary for the model to learn both the nature of the task and the meaning of the newly introduced tokens. We train the system for 20 epochs using batches of 20 elements. The learning rate is initialized to 5×10^{-5} and Adam is employed as the optimizer. The overall architecture consists of 202M parameters, since we fine-tune GPT-2 (117M) alongside the attention mechanism (8M) and the final classifier (77M).

6 Evaluation

To assess the performance of our architecture, we compare it against three different baselines:

Captioning baseline. The first baseline that we consider is a captioning system [Xu *et al.*, 2015]: given an image, this model tries to generate a meaningful description about what it sees. Our goal was to understand the extent to which the textual modality (i.e., the question) was useful to generate answers. The captioning baseline exploits a VGGNet-11 image encoder to extract visual features that are later combined with the outputs of an LSTM. The joint representation then passes through an attention layer, which focuses on specific parts of the image and updates the hidden states of the LSTM. The caption is generated one token at a time and the process goes on until an $<$***eos***$>$ token is produced. We train this model using Teacher Forcing and a loss computed with a doubly stochastic attention regularization that updates the gradient using *CrossEntropy* on the output sequence, with a regularization factor computed starting from the attention maps.

Answering baseline. To check if our architecture was making use of the information contained in the input images, we considered another baseline starting from a pre-trained instance of GPT-2 (Small). We fine-tuned the system to learn how to answer the questions exploiting only the linguistic bias present in the question, without allowing it to condition the generation of the answer on the images.

VQA baseline. For a direct comparison with a standard, publicly-available VQA system, we consider a third baseline [Kazemi and Elqursh, 2017] that uses both question and image to generate the answer. This is a strong baseline specifically designed for VQA tasks, and a sound comparative architecture. The system consists of an image encoder (ResNet-152 [He *et al.*, 2016]), alongside an LSTM which extracts textual features. A compact, but powerful, attention layer brings the two modalities into a common subspace where they are combined to generate attention maps over the image. Finally, the attention output is concatenated back with the LSTM hidden state and the resulting vector is fed to a classifier which distributes probabilities over 3000 possible answers. We do not re-train or fine-tune this architecture, and download the authors' pre-trained instance on the VQAv2 dataset. This architecture follows the standard VQA approach of addressing the task in a classification manner, generating short and concise answers.

6.1 Quantitative Results

We evaluate our models using three different quantitative metrics: Accuracy, BLEU [Papineni *et al.*, 2002], and Word Mover's Distance [Kusner *et al.*, 2015]. We describe each briefly here.

Model	Overall	Yes/No	Number	Other
Captioning	0.00	0.00	0.00	0.00
Answering	0.11	0.20	0.20	0.14
VQA	**0.46**	0.71	**0.27**	**0.41**
VGGPT-2	0.34	**0.73**	0.23	0.24

Table 2: Accuracy scores of the three baselines and VGGPT-2.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Captioning	0.100	0	0.100	0
Answering	0.293	0.149	0.074	0.027
VQA	**0.549**	0.187	0.066	0.011
VGGPT-2	0.463	**0.223**	**0.101**	**0.036**

Table 3: BLEU-k scores of the three baselines and VGGPT-2.

Accuracy. Nearly all existing VQA work reports accuracy as the primary metric, computed as the percentage of times an answer exactly matches *one of* the ground truths. Even though our architecture doesn't generate answers that span across a fixed set of possible choices, we report these results as a comparative means for traditional VQA approaches. It is critical to keep in mind that if a ground truth is, say, "sunny," and our system outputs "it is sunny," the accuracy would be 0. Since our architecture's goal is to generate open-ended answers, this problem occurs very frequently. Nevertheless, we find that our model's accuracy is comparable to that of the VQA baseline (Table 2). Interestingly, when we compare the shortest answers (i.e., binary outputs), VGGPT-2 achieves a higher accuracy than the VQA baseline; it appears that when the model does output short answers, it does so with relatively high precision. For other question types, the VQA baseline performs better, as expected given the preponderance of short, concise answers in both the gold standard and VQA baseline's output.

BLEU. BLEU [Papineni *et al.*, 2002] is commonly used as an evaluation metric for generative language tasks, and thus we incorporated it as an additional evaluation metric here. We compute a corpus-level BLEU score by comparing each output with its respective 10 ground truths from the VQAv2 corpus. In Table 3 we report BLEU-k scores.[4] From these results we highlight a couple interesting findings:

1. As we consider bigger n-gram overlaps (BLEU-2, -3 and -4), VGGPT-2 performs better relative to our VQA baseline. This means that our architecture performs better when it comes to generating longer answers, while the VQA baseline is strongest with single word outputs.
2. The two VQA models outperform the captioning[5] and

[4]When we compute BLEU-k we equally weight n-gram matches from 1 to k to sum up to 1, and set all other weights to 0.

[5]In general, we observe that the captioning baseline failed to properly learn the task, as evidenced by dismal scores for both accuracy and BLEU.

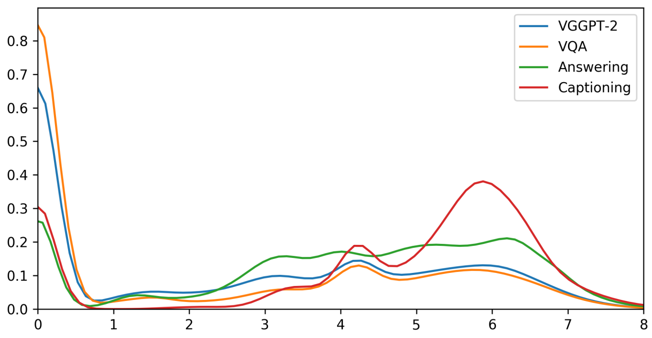


Figure 2: Word Mover's Distance distribution.

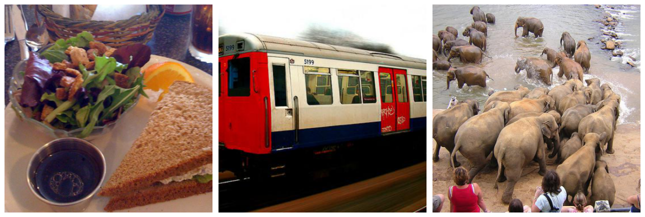
(particularly importantly) answering baselines, confirming that multimodality is key to answer quality.

Word Mover's Distance. To check the semantic similarity between the answers generated by our models and the ground truths, we also evaluated our results using a metric that exploits pre-trained word embeddings, known as Word Mover's Distance (WMD) [Kusner *et al.*, 2015]. We employ WMD with 100-dimensional pre-trained GloVe embeddings [Pennington *et al.*, 2014], finding evidence that VGGPT-2 indeed generates semantically high-quality answers, further supporting the trend of high performance observed with both accuracy and BLEU metrics.

In Figure 2 we plot the WMD distribution for each model. The plot demonstrates that VQA architectures generate a larger amount of answers whose WMD is small (sharper peak around 0), indicating that the latter models are making good use of both the visual and textual modality to generate the outputs. If we compare the VQA baseline with VGGPT-2, we see a strong correlation between the two models since they both tend to produce answers which are semantically close to the ground truths. Overall, the WMD distribution effectively confirms the general trend seen with our other evaluation metrics, and supports the notion that an open-ended, generative architecture offers a feasible alternative to traditional means of addressing the VQA task.

6.2 Qualitative Results

In addition to the quantitative results, we perform a qualitative analysis of our architecture's performance. In Table 4, we compare answers for specific question/image pairs between VGGPT-2 and the VQA baseline. We discover that our model generates, most of the time, longer and correct answers, exploiting the autoregressive behavior of the underlying language model. In contrast, the VQA baseline generates shorter and more concise outputs. This issue becomes more evident when the question requires the output to be descriptive, such as in the case of Picture 1, where the VQA baseline can pick only one of the many objects present in the plate. Picture 2 highlights the need for a language model that allows the system to generate a sequence of terms (in this case colors) to answer correctly. Picture 3 also demonstrates a heightened awareness of the image's events.



(a) Picture 1 (b) Picture 2 (c) Picture 3

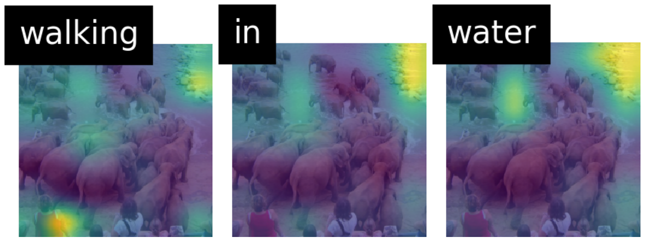


Figure 4: Attention maps associated with Picture 3.

Model	Answer
(a): *What is in the plate?*	
VQA	Salad
VGGPT-2	Toast, orange and salad
(b): *What color is the train?*	
VQA	Red
VGGPT-2	Red and white
(c): *What are the animals doing?*	
VQA	Drinking
VGGPT-2	Walking in water

Table 4: Qualitative results on real-world images.

7 Conclusion

In this work, we proposed a novel Transformer-based architecture for visual question answering that combines textual and visual modalities to generate unconstrained, open-ended answers. We empirically verify the feasibility of leveraging this architecture by framing the problem in a way that diverges from the standard classification perspective, with the objective of building a system that does not distribute probabilities over a fixed set of possible answers but generates them dynamically one token at a time. In future work we plan to experiment with deeper image encoders, such as Residual Networks [He *et al.*, 2016], and will experiment with a multi-head variation of our attention mechanism. We will also examine deeper Transformer architectures in this context, such as the recently released GPT-2 variant with $\approx$ 1.5B parameters. It is our hope that the work reported here stimulates additional interest in open-ended models in the VQA domain, and to that end we also release our model and source code online to facilitate ease of replication.[6]

[6]Link in camera-ready paper.

References

[Anderson *et al.*, 2018] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6077–6086. IEEE, 2018.

[Andreas *et al.*, 2016] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48, 2016.

[Antol *et al.*, 2015] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[Charikar *et al.*, 2002] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

[Fukui *et al.*, 2016] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 457–468, Austin, Texas, November 2016. Association for Computational Linguistics.

[Gao *et al.*, 2016] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 317–326, 2016.

[Goyal *et al.*, 2017] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[Hu *et al.*, 2017] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 804–813, 2017.

[Kazemi and Elqursh, 2017] Vahid Kazemi and Ali Elqursh. Show, ask, attend, and answer: A strong baseline for visual question answering, 2017.

[Kusner *et al.*, 2015] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966, 2015.

[Lamb *et al.*, 2016] Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

[Lu *et al.*, 2016] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297, 2016.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[Shibata *et al.*, 1999] Yusuke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, and Takeshi Shinohara. Byte pair encoding: A text compression scheme that accelerates pattern matching. Technical Report DOI-TR-161, Department of Informatics, Kyushu University, 1999.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Tan and Bansal, 2019] Hao Tan and Mohit Bansal. LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5099–5110, Hong Kong, China, November 2019. Association for Computational Linguistics.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.