MAE-BERT VQA: A Hybrid Model for Visual Question Answering

Junyi Li* Syed Ali Haider†
Department of Computer Science
New York University
Email: jl13122@nyu.edu* sh6070@nyu.edu†

***Abstract*—This project focuses on advancing Visual Question Answering (VQA) by integrating cutting-edge techniques and models. We start by fine-tuning the BLIP-VQA model, incorporating Low-Rank Adaptation (LoRA) layers specifically at the decoder level to enhance model adaptability while maintaining computational efficiency. In parallel, a custom VQA model is developed, using the Masked Autoencoder's (MAE) Vision Transformer (ViT) He, Chen, Xie, *et al.* for image representation and a pre-trained BERT model for text processing. The model architecture is further augmented with a cross-attention mechanism that is first trained for image captioning tasks and subsequently fine-tuned for VQA, ensuring a deep semantic understanding of both visual and textual inputs. This dual-stage training approach allows the model to generate more accurate and contextually relevant answers to visual questions. The project demonstrates significant improvements in VQA performance by combining LoRA-enhanced fine-tuning with sophisticated cross-modal attention strategies, offering a robust framework for future research and application in the field of multimodal AI.**

I. Introduction

Visual Question Answering (VQA) represents a challenging intersection of computer vision and natural language processing, where models must interpret and reason about images in the context of textual queries. Recent advancements in this domain have demonstrated the efficacy of end-to-end trainable models like BLIP (Bootstrapping Language-Image Pre-training),[2] which integrates vision and language processing into a unified architecture. BLIP-VQA, as an end-to-end model, processes images and questions simultaneously, generating accurate answers through a cohesive training approach.

In this project, we enhance the BLIP-VQA framework[2] by implementing Low-Rank Adaptation (LoRA)Hu, Shen, Wallis, *et al.* layers specifically at the decoder stage. This targeted adaptation allows us to fine-tune the model efficiently while leveraging the pretrained capabilities of the BLIP-VQA architecture. By focusing LoRA layers on the decoder, we aim to improve the model's flexibility and performance in generating answers without overhauling the entire system. The use of LoRA in this manner ensures that the adaptation process is computationally efficient and specifically tailored to enhance the decoder's capacity for generating contextually relevant responses.

Our fine-tuning process utilizes the COCO dataset exclusively, focusing solely on training the LoRA layers. COCO, with its diverse and rich annotations, provides a robust foundation for refining the model's performance in real-world scenarios. By restricting the training to LoRA layers, we capitalize on the pretrained BLIP-VQA model's existing knowledge while adapting it to the the specific nuances of the COCO VQA task.[4]

Additionally, our custom VQA model integrates pretrained BERT and Masked Autoencoder (MAE) Vision Transformer (ViT) encoders. The pretrained BERT encoder enriches textual understanding with robust language representations[5], while the MAE ViT encoder provides powerful image features[1]. Combining these encoders with cross-attention mechanisms enables a deep semantic alignment between visual and textual modalities. This approach leverages the strengths of both pretrained models—BERT's nuanced language comprehension and MAE ViT's sophisticated image processing—to enhance the overall performance of the VQA system.

II. Related Work

In the field of Visual Question Answering (VQA)[4], recent advancements in transformer architectures have opened new avenues for improving the interaction between visual and textual data. Key studies in this area lay the groundwork for understanding how vision and language models can be fused effectively for VQA tasks.

Vision Transformers (ViT), introduced by Dosovitskiy, Beyer, Kolesnikov, *et al.* in "An Image is Worth 16x16 Words," [6] replaced convolutional layers with transformer-based architecture for visual tasks, demonstrating its effectiveness for image classification. This architecture breaks images into patches and treats them similarly to word tokens in language models, offering a more flexible and scalable approach to image processing. ViT's patch-based representation allows for more sophisticated handling of visual features, which is crucial for the complex interactions required in VQA tasks.

Building on ViT, He, Chen, Xie, *et al.* in the paper "Masked Autoencoders Are Scalable Vision Learners" [1] introduce the concept of Masked Auto-encoders (MAE) to improve scalability and performance in vision models. MAE masks random image patches and then reconstructs the missing information, forcing the model to focus on high-level semantics. This shift from pixel-level details to semantic features is vital in VQA, where understanding context in images is as important as recognizing objects. MAE's self-supervised approach allows the Vision Transformer to learn robust representations without extensive labeled datasets, making it adaptable to tasks like VQA where labeled data can be sparse.

The "Align before Fuse" [7]paper by Li, Selvaraju, Gotmare, *et al.* improves vision-language interaction through token alignment before fusing them via cross-modal attention. This pre-fusion alignment helps models to better capture nuanced relationships between images and textual inputs, which is pivotal in VQA for reasoning about the image in the context of a given question. Align before Fuse further highlights the importance of pre-aligning visual and textual representations before combining them, a principle leveraged in our project to enhance multi-modal understanding.

Similarly, Li, Li, Xiong, *et al.* in "Bootstrapped Masked Auto-encoders" [2]introduce a momentum encoder and target-aware decoder to pre-train vision BERT models, emphasizing semantic understanding over pixel-based reconstructions. This focus on semantics, applied in both image and textual data, contributes significantly to improved VQA performance as it aids in better reasoning about the image's content in relation to a question. BootMAE's semantic-driven approach aligns well with the objectives of VQA, where understanding both the global context and finer details is necessary for generating accurate answers.

Another crucial development for scaling models in VQA is LoRA (Low-Rank Adaptation of Large Language Models)[3], proposed by Hu, Shen, Wallis, *et al.* LoRA introduces low-rank matrices into existing model parameters, enabling efficient adaptation of large language models to new tasks with a fraction of the trainable parameters. This is particularly relevant for fine-tuning large multi-modal models for specific VQA tasks, where updating the entire model may be computationally prohibitive. In this project, LoRA is applied to the BLIP-VQA model, specifically at the decoder level, to enhance adaptability without increasing the computational footprint. This technique allows us to fine-tune our model efficiently while preserving performance gains from pre-training.

The seminal work by Agrawal, Lu, Antol, *et al.* in "VQA: Visual Question Answering" [4]laid the foundation for the VQA task by introducing a large-scale dataset consisting of open-ended questions about images. Their proposed models integrate convolutional neural networks (CNNs) for visual processing with recurrent neural networks (RNNs) for textual processing. This joint reasoning over image and text inputs marks a significant milestone in VQA development. The VQA dataset has since become a standard benchmark, emphasizing the need for models that can understand complex visual scenes and generate semantically appropriate answers based on both image content and textual questions.

By combining these advancements, this project creates a robust VQA model that incorporates state-of-the-art vision transformers, masked auto-encoders, low-rank adaptation techniques, and cross-modal attention mechanisms. Each of these techniques contributes to improving the model's ability to reason over multi-modal inputs, making our approach particularly effective for the challenges posed by the VQA task. Through this integration of techniques, we aim to push the boundaries of what is possible in VQA, providing a strong framework for future research in multi-modal AI.

III. Dataset Understanding

To get a better understanding of our datasets, we did some preliminary Exploratory Data Analysis. We explored some features to understand how our data is formatted.

The COCO (Common Objects in Context) dataset v1 for Visual Question Answering (VQA) is a large-scale, richly annotated resource widely used for developing and evaluating VQA models. It contains image-question-answer triplets, where each image is paired with diverse types of questions, including yes/no, counting, open-ended, and multiple-choice questions, covering varying levels of complexity. Each question is annotated with multiple human-provided answers (typically 10), allowing for a robust evaluation that accounts for variability in human responses. The dataset's images capture a broad array of everyday scenes featuring multiple objects in diverse contexts, offering a challenging environment for visual understanding. Additionally, the dataset emphasizes balance, particularly in the distribution of answers, to prevent models from exploiting biases, such as skewed occurrences of specific responses like "yes" and "no." COCO-VQA's rich annotations, which include object segmentation, object categories, and bounding boxes, provide valuable supplementary information for models to leverage in comprehending visual scenes. This dataset's large scale, with hundreds of thousands of images and corresponding question-answer pairs, makes it an ideal benchmark for training deep learning models and advancing the field of VQA research. [4]

However, generally speaking, the dataset is relatively small which limited its availability to mostly fine tuning for models. Meanwhile, the answers are relatively short, mainly composed of one word, leading to lack of logical thinking(reasoning) capability of the trained models.

IV. IMPLEMENTATIONS

A. *loRA BLIP VQA*

The 'LoRALayer' class presented here implements a Low-Rank Adaptation (LoRA) layer in PyTorch, a technique used to efficiently fine-tune pre-trained models by introducing trainable low-rank matrices into the original model architecture. The class takes an existing fully connected layer ('original layer') and augments it with two learnable linear transformations, 'lora A' and 'lora B', which project the input through a low-dimensional space of rank 'rank' and back to the original space. This low-rank adaptation is scaled by a factor of 'alpha / rank', controlling the contribution of the added LoRA layer to the original model. The 'forward' method of the 'LoRALayer' computes the output by passing the input through the original layer and adding the scaled result of the low-rank projections, effectively injecting task-specific fine-tuning into the original model while keeping the majority of its parameters frozen. This approach offers computational and memory efficiency, especially in large-scale models, as it significantly reduces the number of parameters to be updated during fine-tuning. Thus, we implemented the lora layers into the decoder blocks of

BLIP VQA model by building a decorator for original feed forward layers, structured as shown below,

B. LoRA Layer Algorithm

Algorithm 1 Initialization of LoRA Layer

```
1: Input: $original\_layer$, $rank$, $alpha$
2: Output: $LoRALayer$ object
3: Steps:
    1) Initialize $original\_layer$ as the original layer to be adapted
    2) Set $rank$ as the rank of the low-rank adaptation
    3) Set $alpha$ as the scaling factor for the low-rank adaptation
    4) Initialize $lora\_A$ as a linear layer with input size $original\_layer.in\_features$ and output size $rank$
    5) Initialize $lora\_B$ as a linear layer with input size $rank$ and output size $original\_layer.out\_features$
    6) Set $scaling$ as the scaling factor for the low-rank adaptation, calculated as $alpha/rank$
```

1) Initialization:

Algorithm 2 Forward Pass of LoRA Layer

```
1: Input: $x$
2: Output: $output$
3: Steps:
    1) Compute $output$ as the sum of:
        • The output of the original layer $original\_layer(x)$
        • The scaled output of the low-rank adaptation $scaling \times lora\_B(lora\_A(x))$
    2) Return $output$
```

2) Forward Pass: $Self.original\_layer$ would keep the original linear layers' parameters as trained by Li, Li, Xiong, *et al.*, we would train only the $self.lora\_A$ and $self.lora\_B$ layers. The rest of parameters of the model would be frozen, remaining unchanged during our finetune.

In Mathematical terms consider a pre-trained layer, $W \in R^{d_{\text{out}} \times d_{\text{in}}}$, where d_{in} is the input dimensionality, and d_{out} is the output dimensionality. LoRA introduces a low-rank decomposition to fine-tune this layer without modifying the original weights. This is done by learning two low-rank matrices $A \in R^{r \times d_{\text{in}}}$ and $B \in R^{d_{\text{out}} \times r}$, where $r \ll \min(d_{\text{in}}, d_{\text{out}})$.

The output of the modified layer is:

$$\hat{y} = Wx + \frac{\alpha}{r} BAx$$

Where:

- $x \in R^{d_{\text{in}}}$ is the input to the layer,
- $W \in R^{d_{\text{out}} \times d_{\text{in}}}$ is the original layer's weight matrix,
- $A \in R^{r \times d_{\text{in}}}$ and $B \in R^{d_{\text{out}} \times r}$ are the learned low-rank matrices,
- α is a scaling factor that controls the impact of the low-rank adaptation,
- r is the rank, controlling the bottleneck of the adaptation.

This formulation allows fine-tuning of the network with fewer parameters by learning only the low-rank updates while maintaining the original model's pre-trained weights intact.

C. Modified Decoder Struture

The ‘loraBertLMHeadModel‘ class extends ‘BertPreTrainedModel‘ and introduces several modifications to integrate Low-Rank Adaptation (LoRA) into a BERT-based language model. The primary change is the substitution of the standard BERT model with a ‘loraBertModel‘ in the initialization method, which incorporates LoRA techniques to adaptively adjust the model's capacity. This adaptation is achieved by using ‘loraBertModel‘ instead of ‘BertModel‘, while maintaining the original MLM head (‘self.cls‘) for language modeling. The ‘forward‘ method remains largely unchanged but now utilizes the LoRA-enhanced BERT model to compute sequence outputs and prediction scores. The ‘prepare inputs for generation‘ and ‘reorder cache‘ methods, essential for text generation and beam search, are preserved to ensure compatibility with the new model architecture. These modifications enable the model to leverage LoRA for efficient fine-tuning while retaining the core functionality of BERT for language modeling tasks.

D. Proposed Structure

1) MAE Backbones: Pretrained Masked Auto Encoders

a) Pretrained MAEs for Feature Extraction:

BERT for Text Embeddings: Utilize a pretrained BERT model to convert text (questions) into high-quality contextual embeddings. BERT's bidirectional attention mechanism helps capture rich contextual information from the questions. ViT for Image Embeddings: Employ a pretrained Vision Transformer (ViT) to process images and extract visual features. ViT's transformer-based architecture is adept at capturing global image features and context.

b) Encoder-Decoder Architecture with Cross-Attention: Encoder: Text Encoder: The BERT embeddings of the questions are passed through an encoder layer. This encoder processes the text embeddings and captures the semantic meaning of the questions. Image Encoder: The visual embeddings from ViT are processed through a separate encoder layer. This encoder captures visual features and context from the images. Cross-Attention Mechanism: Implement cross-attention blocks that allow the model to attend to both the text and visual embeddings. This mechanism facilitates the integration of visual and textual information, enabling the model to generate answers based on both sources of input. Decoder: Use a decoder equipped with cross-attention layers that attend to the combined text and visual embeddings to generate the final answer. The decoder processes the integrated information and outputs the answer in natural language.

2) Fusion : a) Feature Extraction:
Text Processing: Pass the input question through the pretrained BERT model to obtain text embeddings. This step involves tokenizing the text and generating embeddings for each token. Image Processing: Feed the input image through the pretrained ViT model to obtain image embeddings. The image is split into patches, and the ViT model generates embeddings for each patch.

b) Integration with Cross-Attention: Text-to-Image Cross-Attention: Implement cross-attention layers in the encoder to align text embeddings with image embeddings. This allows the model to focus on relevant visual features corresponding to different parts of the text. Image-to-Text Cross-Attention: Similarly, implement cross-attention layers in the decoder to integrate visual features with text context. This helps in generating answers that are coherent with both the question and the visual content.

c) Answer Generation: The decoder generates answers based on the integrated text and visual features. The answer generation process involves leveraging the attention mechanisms to ensure that the generated response is contextually appropriate and relevant to both the question and the image.

V. Evaluation

A. Methods

The accuracy of the Visual Question Answering (VQA) model is evaluated by comparing predicted answers to ground truth answers based on their semantic similarity rather than exact matches. This approach accommodates variations in phrasing and synonyms. The accuracy calculation is performed using the following steps:

1. Answer Normalization: The predicted and ground truth answers are first normalized to ensure consistent comparison. The normalization process involves:

- Converting the text to lowercase.
- Trimming leading and trailing whitespace.
- Removing punctuation.

2. Embedding Generation: Both predicted and ground truth answers are converted into embeddings using a BERT model. The embedding generation involves:

- Tokenizing the input text and generating BERT embeddings.
- Averaging the token embeddings to obtain a single vector representation for each answer.

The function for generating embeddings is as follows:

Algorithm 3 Get Embedding

```
1: function get_embedding(text)
2:     tokens ← tokenizer(text, return_tensors='pt')
3:     with torch.no_grad():
4:         outputs ← model(**tokens)
5:     embeddings ← outputs.last_hidden_state.mean(dim=1)

6:     return embeddings.numpy()
7: end function
```

3. Cosine Similarity Calculation: The cosine similarity between the embeddings of the predicted and ground truth answers is computed to measure their semantic similarity. The cosine similarity between two vectors $\mathbf{t}$ and $\mathbf{e}$ is calculated using the formula:

$$\cos(t,e) = \frac{\mathbf{t}\cdot\mathbf{e}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} t_i e_i}{\sqrt{\sum_{i=1}^{n}(t_i)^2}\sqrt{\sum_{i=1}^{n}(e_i)^2}}$$

where t_i and e_i are the components of vectors $\mathbf{t}$ and $\mathbf{e}$ respectively. $\sum_{i=1}^{n} t_i e_i$ denotes the dot product of vectors $\mathbf{t}$ and $\mathbf{e}$. $\sqrt{\sum_{i=1}^{n}(t_i)^2}$ and $\sqrt{\sum_{i=1}^{n}(e_i)^2}$ represent the magnitudes (Euclidean norms) of vectors $\mathbf{t}$ and $\mathbf{e}$ respectively.

This formula computes the cosine of the angle between the two vectors, providing a value between -1 and 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates perfect dissimilarity.

4. Correctness Evaluation: An answer is considered correct if the cosine similarity between its embedding and the embedding of the ground truth answer exceeds a predefined threshold (e.g., 0.8). The correctness check is defined as:

Algorithm 4 Check if Predicted Answer is Correct

```
1: Input: $pred$, $gt$, $threshold = 0.8$
2: Output: is_correct
3: Normalize the predicted answer: $pred$ ← normalize_answer($pred$)
4: Normalize the ground truth answer: $gt$ ← normalize_answer($gt$)
5: Calculate similarity: $similarity$ ← cosine_similarity_score($pred, gt$)
6: return $similarity > threshold$
```

5. Accuracy Calculation: The overall accuracy is computed by iterating through all predicted answers, ground truth answers, and associated questions. For each answer, the code prints the question, predicted answer, and ground truth answer, and checks if the predicted answer is correct. The accuracy is then calculated as:

Algorithm 5 Calculate Accuracy

```
1: Input: $all\_preds$, $all\_labels$, $all\_questions$
2: Output: $accuracy$
3: $correct\_predictions \leftarrow 0$
4: for (question, pred, label) in zip(all_questions, all_preds, all_labels) do
5:     Print "question:", question
6:     Print "pred:", pred
7:     Print "label:", label
8:     if is_answer_correct(pred, label) then
9:         $correct\_predictions \leftarrow correct\_predictions + 1$
10:    end if
11: end for
12: return $\frac{correct\_predictions}{\text{len}(all\_preds)}$
```

'correct predictions' is the count of answers deemed correct, and the accuracy is the ratio of correct predictions to the total number of predictions. The accuracy calculation method described allows for evaluating the performance of a VQA model based on semantic similarity, accommodating variations in answer phrasing and ensuring a more flexible and accurate assessment of the model's performance. By using this mechanism,

B. Results

1) LoRA BLIP VQA: Data

With this accuracy mechanism, the model achieves 83.57% acccuracy on COCO validation dataset while the model fine-tuned with LoRA implemented achieved an acccuracy of 85.94%.

Paculiarities

1. Accuracy on Common and Simple Queries: The model performs well on straightforward and common queries, such as identifying objects (”what kind of object is the kid riding?”), basic yes/no questions (”is this a hospital?”), and simple counting tasks (”how many cats are in the image?”). These results indicate that the model has a solid understanding of basic visual features and is capable of associating them with corresponding text-based queries.

2. Handling of Specific Object and Scene Recognition: The model accurately identifies specific objects and scenes, such as ”snowboard,” ”train,” ”church,” and ”tennis.” This shows that the model has been effectively trained on a diverse dataset, allowing it to recognize various objects and scenes across different contexts.

3. Correct Interpretation of Some Visual Contexts: In instances where the model is required to understand context or relationships between objects (e.g., identifying the number of people, recognizing that a person is snowboarding), it demonstrates an understanding of the visual context, especially when the context is clear and unambiguous.

4. Robustness in Answering Binary Questions: The model generally performs well on yes/no questions, which are typically easier for AI models because they require binary decisions. This suggests that the model is good at identifying the presence or absence of certain features in images.

5. Struggles with Ambiguity and Contextual Nuances: The model occasionally misinterprets questions that require nuanced understanding or interpretation of context. For instance, it incorrectly answered ”who is in front of the cake with candles?” by labeling ”boy” instead of the correct ”woman.” This indicates that the model may struggle with identifying specific roles or actions in a complex scene.

6. Errors in Counting and Spatial Understanding: The model makes mistakes in counting objects, such as in ”how many stories is the building on the left?” (predicted 5, labeled 3) and ”how many people can you see in the picture?” (predicted 7, labeled 8). This highlights a limitation in the model's spatial reasoning and ability to accurately count or differentiate multiple objects in a scene. Notably, LoRA BLIP VQA performed better at calculation of objects, though it still could not gurantee 100% accuracy.

7. Inconsistent Performance on Fine-Grained Details: The model sometimes fails to capture fine-grained details, such as ”what time is it on the clock?” (predicted 6, labeled 8:30) or ”what pattern is painted on the helmet?” (predicted solid, labeled no helmet). This suggests that the model's attention mechanism might not be sufficiently fine-tuned to focus on small, specific details in an image.

8. Inaccuracies in Complex Visual and Textual Association: The model struggles with questions that require a deeper association between visual elements and the corresponding text. For example, it mistook ”what is cast?” (predicted shadow, labeled shadows) and ”where is the bird?” (predicted tire, labeled in tree). These errors indicate challenges in accurately associating visual details with complex textual queries.

9. Overconfidence in Incorrect Answers: The model shows a tendency to provide confident but incorrect answers. For instance, in questions like ”what year is the car?” (predicted 2000, labeled 2014), the model confidently produces an answer, but it's incorrect. This overconfidence may stem from a lack of uncertainty quantification in the model's predictions.

10. Difficulty with Generalization: The model has trouble generalizing in cases where it encounters unfamiliar or less common scenarios, such as interpreting what is happening in a scene or correctly identifying abstract elements. This limitation suggests that the model's training data may not have sufficiently covered all possible scenarios or that the model's architecture is not fully capable of generalizing from the data it has seen.

C. Failure in Training of the New Model

In this section, we showcase and evaluate the predictions generated by our model on both the Train2014 and Eval sets. The following examples illustrate the responses provided by the model to various questions after training for 16 epochs:

- **Image ID: 446870**
  - **Question:** Are these chocolate doughnuts?
  - **Generated Answer:** cheese cheese ... cheese
- **Image ID: 529630**
  - **Question:** How many different dishes can you see?
  - **Generated Answer:** is is is ... is
- **Image ID: 462987**
  - **Question:** Is that a bookshelf?
  - **Generated Answer:** is is is ... is
- **Image ID: 462046**
  - **Question:** Are both skiers skiing in another person's tracks?
  - **Generated Answer:** is is ... is
- **Image ID: 402542**
  - **Question:** What color is the wall?
  - **Generated Answer:** street street street street ... street

D. Model Predictions on Eval Set

- **Image ID: 458752**
  - **Question:** What is this photo taken looking through?
  - **Generated Answer:** is is ... is

The evaluation of the generated results reveals that the model's predictions are often repetitive and lack meaningful content. For instance, responses such as *"cheese cheese cheese ..."* and *"is is is ..."* indicate that the model may be struggling to provide relevant answers to the questions asked. This issue is prominent across both the Train2014 and Eval sets.

Several factors might contribute to these suboptimal results:

- **Dataset Size:** The COCO dataset, while diverse, is relatively small in scale compared to larger datasets. The model may have overfitted to the training data and might not generalize well to new examples.
- **Model Convergence:** It is possible that the model has converged prematurely, limiting its ability to generate varied and contextually appropriate responses. Further fine-tuning or training with more diverse data might be necessary.
- **Model Capacity:** The model's capacity might be insufficient for the complexity of the task. Increasing the model size or exploring advanced architectures could improve performance.
- **Data Quality:** There might be issues with the quality or relevance of the training data, impacting the model's ability to learn meaningful patterns.

While the model shows some capability in generating responses, there is significant room for improvement. Future work will focus on addressing these issues by expanding the dataset, adjusting model parameters, and exploring more advanced model architectures.

VI. Conclusion

In this work, we explored the integration of Low-Rank Adaptation (LoRA) layers into the BLIP VQA model and evaluated its performance on the COCO dataset. The implementation of LoRA layers demonstrated promising results, with an improvement in accuracy from 83.57% to 85.94% on the COCO validation dataset. This enhancement underscores the effectiveness of LoRA in fine-tuning large pre-trained models with reduced computational and memory costs.

The evaluation revealed several strengths and limitations of the LoRA-enhanced BLIP VQA model. The model performed well on straightforward queries and demonstrated robust capabilities in object and scene recognition. However, challenges remain in handling ambiguous contexts, counting objects accurately, and capturing fine-grained details. The model also exhibited a tendency for overconfidence in its predictions and difficulties in generalizing to less common scenarios.

The analysis of the model's failure during the training phase indicated several areas for improvement. Issues such as repetitive responses and lack of meaningful content suggest potential problems with dataset size, model convergence, capacity, and data quality. To address these challenges, future work will focus on expanding the dataset to include more diverse examples, adjusting model parameters for better convergence, and exploring advanced architectures to enhance model capacity and generalization.

Overall, while the current results are encouraging, there is significant potential for improvement. Continued refinement of the model and training procedures is necessary to achieve more accurate and contextually relevant responses. Future research will aim to address the identified limitations and explore new methodologies to advance the state-of-the-art in Visual Question Answering.

References

[1] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, *Masked autoencoders are scalable vision learners*, 2021. arXiv: 2111.06377 `[cs.CV]`. [Online]. Available: https://arxiv.org/abs/2111.06377.

[2] J. Li, D. Li, C. Xiong, and S. Hoi, *Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation*, 2022. arXiv: 2201.12086 `[cs.CV]`. [Online]. Available: https://arxiv.org/abs/2201.12086.

[3] E. J. Hu, Y. Shen, P. Wallis, *et al.*, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 `[cs.CL]`. [Online]. Available: https://arxiv.org/abs/2106.09685.

[4] A. Agrawal, J. Lu, S. Antol, *et al.*, *Vqa: Visual question answering*, 2016. arXiv: 1505.00468 `[cs.CL]`. [Online]. Available: https://arxiv.org/abs/1505.00468.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 `[cs.CL]`. [Online]. Available: https://arxiv.org/abs/1810.04805.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: 2010.11929 `[cs.CV]`. [Online]. Available: https://arxiv.org/abs/2010.11929.

[7] J. Li, R. R. Selvaraju, A. D. Gotmare, S. Joty, C. Xiong, and S. Hoi, *Align before fuse: Vision and language representation learning with momentum distillation*, 2021. arXiv: 2107.07651 `[cs.CV]`. [Online]. Available: https://arxiv.org/abs/2107.07651.