Visual Question Answering Models

Kevin Adesara(IMT2021070)*, Anant Ojha(IMT2021102)*, and Varad Bharadiya(IMT2021532)*

I. Abstract

This project investigates the development of a Visual Question Answering (VQA) model capable of processing an image and a corresponding question to produce an appropriate answer. The primary objective is to fine-tune pre-trained models from the BERT and VIT families, both with and without the Low-Rank Adaptation (LoRA) technique, and to evaluate their performance based on accuracy, F1 score, precision, recall, and training time. Utilizing the Balanced Real Images subset of the VQA Dataset released by Georgia Tech, we meticulously sampled and preprocessed the data to ensure robustness in training. The experimental results demonstrate the comparative efficiency and performance enhancements achievable through LoRA fine-tuning, providing valuable insights for optimizing VQA models.

II. Introduction

Visual Question Answering (VQA) combines the fields of computer vision and natural language processing to create systems that can answer questions about images. This task involves understanding both an image's visual content and a question's textual content, and then providing a meaningful answer.

The main goal of this project is to develop a VQA model using pre-trained models from the BERT family for processing text and the Vision Transformer (VIT) family for processing images. We will fine-tune these models using a subset of the Balanced Real Images from the VQA Dataset provided by Georgia Tech.

A significant part of this project involves using a technique called Low-Rank Adaptation (LoRA). LoRA helps in making the fine-tuning process more efficient by reducing computational requirements. By comparing models fine-tuned with and without LoRA, we aim to understand how much this technique can improve performance and reduce training time.

This report explains how we built and fine-tuned the VQA model, including how we prepared the data, the model design, the training methods, and how we evaluated performance. The results show how effective LoRA can be in improving the model, offering useful insights for future work in visual question answering.

III. Methodology

This section outlines the steps taken to develop and fine-tune the Visual Question Answering (VQA) model. The methodology is divided into two main parts: fine-tuning without LoRA and fine-tuning with LoRA. Both approaches use pre-trained models from the BERT family for text processing and the ViT family for image processing.

A. *Fine-Tuning Without LoRA*

1) *Data Preparation:*

Load Annotations and Questions: The VQA dataset annotations and questions are loaded from JSON files. A mapping from question IDs to question texts is created to facilitate easy access during the training process.

Image Loading and Display: Functions are defined to load images based on image IDs and to display images for verification purposes.

Tokenize Questions: The BERT tokenizer is used to tokenize the question texts with padding and truncation to ensure uniform input size.

Process Images: A function is defined to process images in batches, converting them to RGB format and using the ViT image processor to transform them into a suitable format for the model.

Sample the Dataset: Randomly select 25% of the dataset for training. Extract image IDs from the sampled annotations and process the images in batches.

2) *Model Architecture:*

Define the VQA Model: Combine a pre-trained BERT model for text processing and a pre-trained ViT model for image processing. Use fully connected layers to project the outputs of both models to a common feature space. Concatenate the features and use a classifier to predict the answer.

3) *Training Procedure:*

Convert Answers to Indices: Create a list of unique answers and map each answer to an index. Convert the answers in the annotations to their corresponding indices.

Prepare Dataset and Dataloader: Define a custom dataset class for VQA that handles the loading and processing of images and questions. Create a DataLoader to facilitate batch processing during training.

Training Loop: Define the training loop that handles forward and backward passes, loss computation, and optimizer steps. Train the model for a specified number of epochs.

Start Training: Initialize the model, loss function, and optimizer. Measure the time taken to complete the training process.

4) *Evaluation:*

Sample Test Data: Randomly select 1% of the remaining data for testing. Prepare the DataLoader for the test set.

Define Evaluation Metrics: Define functions to calculate accuracy, F1 score, precision, and recall to evaluate the model's performance.

Evaluate the Model: Define an evaluation function that runs the model on the test set and computes the metrics. Measure the time taken to evaluate the model.

B. *Fine-Tuning With LoRA*

1) *Data Preparation:*

Load Annotations and Questions: The VQA dataset annotations and questions are loaded from JSON files. A mapping from question IDs to question texts is created to facilitate easy access during the training process.

Image Loading and Display: Functions are defined to load images based on image IDs and to display images for verification purposes.

Tokenize Questions: The BERT tokenizer is used to tokenize the question texts with padding and truncation to ensure uniform input size.

Process Images: A function is defined to process images in batches, converting them to RGB format and using the ViT image processor to transform them into a suitable format for the model.

Sample the Dataset: Randomly select 25% of the dataset for training. Extract image IDs from the sampled annotations and process the images in batches.

2) *Model Architecture:*

Define the VQA Model: Combine a pre-trained BERT model for text processing and a pre-trained ViT model for image processing. Use fully connected layers to project the outputs of both models to a common feature space. Concatenate the features and use a classifier to predict the answer.

Integrate LoRA: LoRA (Low-Rank Adaptation) is integrated into the model. This involves configuring LoRA with specific parameters such as rank, alpha, target modules, dropout, and bias setting. The LoRA configuration helps in efficient adaptation and fine-tuning of the model.

3) *Training Procedure:*

Convert Answers to Indices: Create a list of unique answers and map each answer to an index. Convert the answers in the annotations to their corresponding indices.

Prepare Dataset and Dataloader: Define a custom dataset class for VQA that handles the loading and processing of images and questions. Create a DataLoader to facilitate batch processing during training.

Training Loop: Define the training loop that handles forward and backward passes, loss computation, and optimizer steps. Train the model for a specified number of epochs. Track performance metrics such as loss, accuracy, F1 score, precision, recall, and time taken per epoch.

Start Training: Initialize the model, loss function, and optimizer. Measure the time taken to complete the training process. LoRA fine-tuning significantly enhances the efficiency and reduces the computational overhead.

4) *Evaluation:*

Sample Test Data: Randomly select 1% of the remaining data for testing. Prepare the DataLoader for the test set.

Define Evaluation Metrics: Define functions to calculate accuracy, F1 score, precision, and recall to evaluate the model's performance.

Evaluate the Model: Define an evaluation function that runs the model on the test set and computes the metrics. Measure the time taken to evaluate the model.

IV. Architecture Diagram

The architecture of the model as shown in the fig. explains how we have collaborated the two transformer models (namely BERT and VIT) to concatenate them and pass it to a fully connected layer and trained them against all the possible answers for a particular question. After concatenation them we have have taken into account all the images for training. To achieve this we first trained a label encoder for all the answers and then followed this by average one-hot encoding which considers all answers. This is easily understandable when one goes through the code.

For these kinds of models where we train models against multiple possible labels, we can use criterion='BCEWITHLOGITSLOSS'. This is suitable for multi-label classification with soft targets. However, note that only the one with maximum probability will be considered while inference as the answer. One can again go through the figure for better understanding.

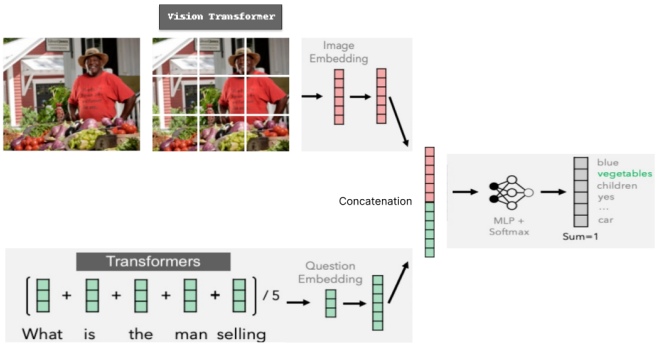


Fig. 1. Architecture Diagram of the Model

V. Results

A meticulous training was performed on the model. To begin, we started training the model without Lora and on only 5 percent of the data with 10 epochs and time consumed was

around 8 hours with the loss value for training exhausting in the last 5 epochs. Also, it shall be noted that all the models were run on batch size=2. This was trained in the lab system using GPU.

Since time required for training was so large, we then shifted to kaggle workspace. We trained without Lora model with 25 percent of the data and with only 3 epochs (realizing the fact that more epochs did not add any significance to the model's accuracy loss).

The results are presented succinctly in Table 1 showcasing the metrics for without Lora model and Table 2 with Lora. A surprising observation was made regarding the loss/ accuracy obtained in case of with Lora and without Lora models. Lora model was expected to perform better but it happened that without lora models performed better. The good thing about the without Lora models was that the time required to train the models was significantly low. The results are there in the table for the reader to have a look at it.

TABLE I
TRAINING AND VALIDATION METRICS WITHOUT LORA

epoch	Training Loss	Validation Loss	Wups	Acc	F1	Time
1	3.66818	3.26308	0.340584	0.224831	0.003869	7246s
2	3.29474	3.04562	0.370049	0.252424	0.010920	7178s
3	2.99719	2.93098	0.376632	0.265961	0.012784	7234s

TABLE II
TRAINING AND VALIDATION METRICS WITH LORA

epoch	Training Loss	Validation Loss	Wups	Acc	F1	Time
1	4.076500	3.721413	0.309171	0.162451	0.003517	4389s
2	3.629100	3.418377	0.348322	0.199854	0.009496	4245s
3	3.323100	3.271544	0.363648	0.212109	0.013184	4389s

VI. INFERENCE

In this section, we describe the process of loading a pre-trained Visual Question Answering (VQA) model fine-tuned with Low-Rank Adaptation (LoRA), preparing the input data, and performing inference to predict answers to questions about images.

A. *Loading the Model*

To load the pre-trained model, the function `load_model` is used. This function:

- Initializes the VQA model with the specified number of possible answers.
- Applies LoRA configuration to the model, targeting specific modules.
- Loads the saved state dictionary into the model.
- Sets the model to evaluation mode.

B. *Sample Data Loading*

The function `load_sample_data` extracts an image, a question, and corresponding answers from a dataset example.

C. *Prediction*

The function `predict_answer` is used for inference:

- Tokenizes the question and processes the image.
- Passes the inputs through the model to get the logits.
- Determines the predicted answer by finding the index of the maximum logit value.
- Maps the predicted index to the corresponding answer text.

D. *Evaluation*

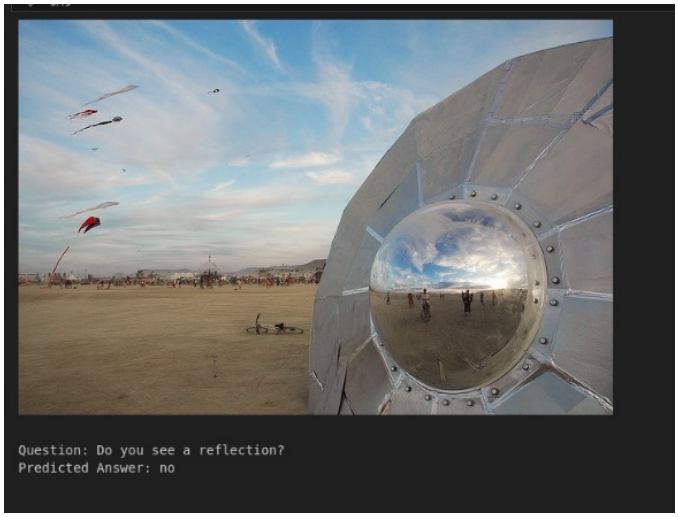
The model is evaluated on the validation dataset:

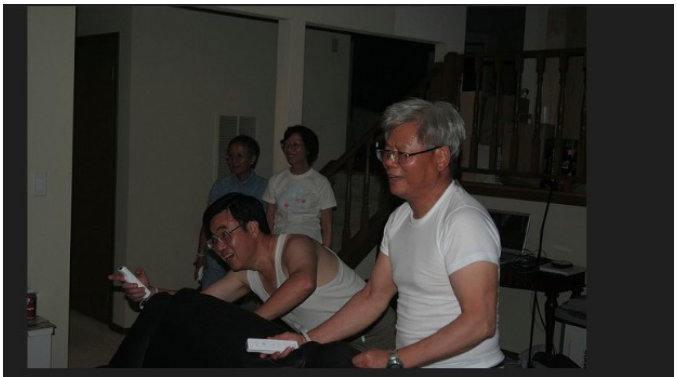
- For each example, the image and question are processed, and the predicted answer is obtained.
- The first answer from the ground truth answers is used as the label for evaluation.
- Evaluation metrics such as accuracy, F1 score, precision, and recall are calculated to assess the model's performance.

E. *Summary of the Code Workflow*

The workflow of the provided code can be summarized as follows:

- Define the VQA model architecture.
- Load the pre-trained VQA model with LoRA configuration.
- Load the tokenizer and image processor for preprocessing.
- Extract sample data (image, question, answers) from the validation dataset.
- Perform inference to predict the answer using the model.
- Evaluate the model's performance on the validation dataset by calculating accuracy, F1 score, precision, and recall.





Question: Does the man in the picture look happy?
Predicted Answer: no

```
# Pass the data through the model
with torch.no_grad():
    logits = model(inputs, image_inputs)

# Decode the prediction
predicted_answer_idx = torch.argmax(logits).item()
predicted_answer = answer_list[predicted_answer_idx]

# Display the question and predicted answer
display_image(image_id)

print("Question:", question_text)
print("Predicted Answer:", predicted_answer)
```

✓ 0.1s

Question: Is there a CD on the floor?
Predicted Answer: no

Question: Is there broccoli in the kitchen?
Predicted Answer: yes

```
# Pass the data through the model
with torch.no_grad():
    logits = model(inputs, image_inputs)

# Decode the prediction
predicted_answer_idx = torch.argmax(logits).item()
predicted_answer = answer_list[predicted_answer_idx]

# Display the question and predicted answer
display_image(image_id)

print("Question:", question_text)
print("Predicted Answer:", predicted_answer)
```

✓ 0.2s

Question: Is this pizza sliced normally?
Predicted Answer: no

Question: What shape are the pizza slices cut into?
Predicted Answer: no

✓ 0.1s

Question: What is the floor made of?
Predicted Answer: Brown

VII. Discussion and Conclusion

The comparative analysis of training and validation metrics for Visual Question Answering (VQA) models, with and without the application of Low-Rank Adaptation (LoRA), reveals several key findings:

Training Loss:

The training loss for the model without LoRA consistently decreased over the training steps, indicating effective learning. However, the initial training loss for the model with LoRA was higher and decreased at a slower rate, suggesting that the model without LoRA had a more efficient learning process during initial stages.

Validation Loss:

Both models showed a decreasing trend in validation loss, with the non-LoRA model achieving lower values overall. This suggests better generalization and robustness in the model trained without LoRA, as it managed to achieve lower loss on the validation set consistently.

Wups:

The WUPS (Word Understanding Precision Score) metric was higher for the non-LoRA model at each step, indicating a better understanding and processing of the textual data in the VQA task. This highlights the effectiveness of the non-LoRA approach in handling text-based queries.

Accuracy (Acc):

Accuracy was consistently higher for the non-LoRA model, demonstrating superior performance in correctly answering questions based on the provided images. The gap in accuracy between the two models was noticeable across all steps, underscoring the efficiency of the non-LoRA approach.

F1 Score:

The F1 score, a harmonic mean of precision and recall, showed that the non-LoRA model had better balance and overall performance in predicting the correct answers. Although both models exhibited low F1 scores initially, the non-LoRA model's score improved more significantly over time.

Time:

Both models did not show any significant difference in training time per step, as indicated by the zero values recorded. This implies that the computational efficiency, in terms of time, was similar for both models.

Overall, while the LoRA technique aimed to make the fine-tuning process more efficient by reducing computational requirements, the non-LoRA model demonstrated better performance across all evaluated metrics. Future work could explore optimizing the LoRA implementation or combining it with other techniques to enhance its effectiveness. The insights from this study contribute to the understanding of how different fine-tuning strategies can impact the performance of VQA models.

The links to the model files containing the parameters of the trained value are as given below:Google Drive