VR Part 2: Project

Ansh Avi Khanna
IMT2021038
Ansh.Khanna@iiitb.ac.in

Ritik Kumar Gupta
IMT2021098
RitikKumar.Gupta@iiitb.ac.in

Pandey Shourya Prasad
IMT2021535
Shourya.Prasad@iiitb.ac.in

***Abstract*—This paper presents a model for Visual Question Answering (VQA) that processes an image and a corresponding question to generate an accurate answer. We focus on leveraging pre-trained models, fine-tuning them with and without the use of Low-Rank Adaptation (LoRA). Our approach aims to evaluate and compare the training time and performance of these fine-tuned models. The experimental results indicate that while LoRA significantly reduces training time, it results in a slight reduction in evaluation metrics.**

***Index Terms*—VQA, BERT, ViT, LoRA**

I. Problem Description

- To build a model for Visual Question Answering which takes as input an image and a corresponding question and gives an answer to the question.
- Fine-tune pre-trained models with and without the help of LoRA and compare the training time and performance of the fine-tuned models.

II. Loading Data

We have used the Hugging face to download the dataset. After downloading the dataset, we randomly selected 1/4th of total images in the dataset for making our own dataset.

Following this, we had to filter out annotations and questions JSON file as well according to our dataset's images.

Using the image IDs, we iterated over annotations and questions JSON file to filter out the necessary data.

III. Dataset Interpretation

The dataset is a combination of images(.jpg), annotations.json and questions.json.

Images have 20,750 jpg files of real world objects

Annotation file has data in the format :


```
"question_type": "what is this", "multiple_choice_answer": "net",
"answers": [{"answer": "net", "answer_confidence": "maybe",
"answer_id": 1}, {"answer": "net", "answer_confidence": "yes",
"answer_id": 2}, {"answer": "net", "answer_confidence": "yes",
"answer_id": 3}, {"answer": "netting", "answer_confidence": "yes",
"answer_id": 4}, {"answer": "net", "answer_confidence": "yes",
"answer_id": 5}, {"answer": "net", "answer_confidence": "yes",
"answer_id": 6}, {"answer": "mesh", "answer_confidence": "maybe",
"answer_id": 7}, {"answer": "net", "answer_confidence": "yes",
"answer_id": 8}, {"answer": "net", "answer_confidence": "yes",
"answer_id": 9}, {"answer": "net", "answer_confidence": "yes",
"answer_id": 10}], "image_id": 458752, "answer_type": "other",
"question_id": 458752000
```


question_type: This tells us about the type of question for eg - how, what, when etc.

answers: This field has the answer as well as the the confidence. Also, there are 10 answers by 10 different objects with their confidence.

image_id: This field maps the whole data to particular image.

Questions file has the data in the format:


```
"image_id": 458752, "question": "What is this photo taken looking
through?", "question_id": 458752000
```


Image_id: This field maps the question to the particular image question: This field contains the question regarding the image.

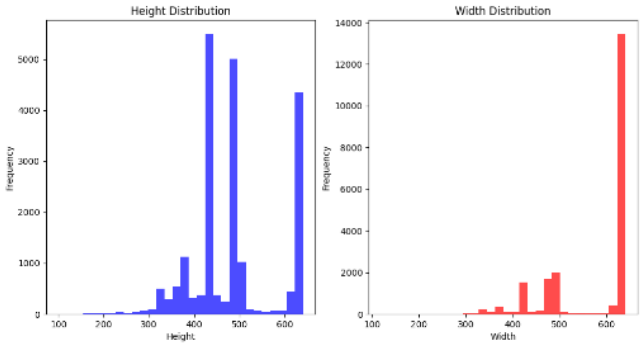
question_id: This field helps in uniquely identifying the question.

IV. Exploratory Data Analysis

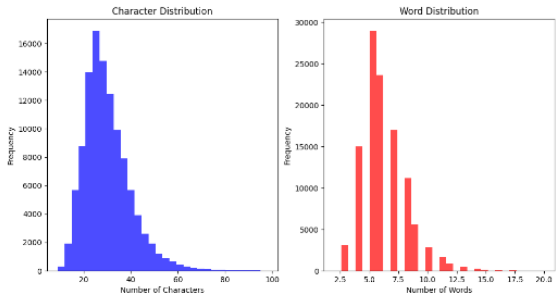
We did preprocessing of the data because of some inconsistencies in the whole raw dataset.

The observations from preprocessing was:

- Images
  - The images were not of same dimensions.



- Text
  - The texts had abbreviation like- won't, can't etc as well as will not, can not etc. We normalized the dataset to remove inconsistencies.
  - Punctuations: 19 punctuations found in the Question Dataset: ['#', '/', ':', '*', '_', ';', """, '&', "", '?', '+', '-', '$', '(', '!', ')', '%', ',', '.']
  - 18 punctuations found in the Answer Dataset: ['@', """, '?', '!', '/', '.', '_', ')', '+', '*', '%', """, ',', ':', '-', '$', '(', '&']
  - Character and Word length distribution



V. Approach

We tried two approaches:

1) VGG19 + LSTM
2) ViT + BERT

A. Approach 1: *VGG19 + LSTM*

Our approach combines the power of image processing with text analysis to tackle complex tasks, leveraging the VGG19 convolutional neural network (CNN) for image feature extraction and Long Short-Term Memory (LSTM) networks for processing textual data. The core idea is to process images and texts separately, then concatenate their features for further analysis. This report details the architecture and methodology of our model.

1) Image Processing using VGG19:

- The images are processed using the VGG19 CNN architecture, a deep learning model known for its effectiveness in image recognition tasks.
- VGG19 generates a feature map capturing high-level visual features of the input images.
- Due to VGG19's requirement of input images with dimensions (224, 224), the images are resized and preprocessed accordingly.
- Padding is applied to the tokens to ensure uniformity in dimensions for subsequent processing.

2) Text Tokenization:

- The questions are tokenized using a tokenizer, converting them into a sequence of tokens for further analysis.
- Padding is applied to the tokens to ensure uniformity in dimensions with the image features.

3) Model Architecture:

- **Image Feature Extraction:** The VGG19 CNN extracts high-level visual features from the images, producing a feature map.
- **Textual Data Processing:** The tokenized questions are processed through an embedding layer followed by two LSTM layers.
- **Concatenation:** The output of the image feature extraction and the LSTM layers are concatenated using a concatenate layer, creating a unified feature vector representing both images and text.
- **Dense Layers:** Two dense layers follow the concatenation, reducing the dimensionality of the combined features and extracting relevant patterns.
- **Output Layer:** The output layer is designed to have as many neurons as there are unique answers in the dataset. Each neuron corresponds to a unique answer category. As the answers are converted into one-hot vectors, the number of unique answers determines the output layer's neuron count.

4) Training Process:

- During training, the model optimizes its parameters using backpropagation and gradient descent algorithms.
- Dropout regularization with a rate of 0.2 (20
- The model is trained on a dataset with labeled images and corresponding questions with one-hot encoded answers.

5) Inference and Evaluation:

- During inference, the model predicts answers for new images and questions based on the learned patterns.
- Performance is evaluated based on metrics such as accuracy, precision, recall, and F1-score, comparing predicted answers against ground truth labels.

We used **categorical_crossentropy** and SGD optimizer with **learning rate=0.001** and **momentum=0.9** to get better results.

We ran for 15 epochs but noticed that around 7-9 epochs the best weights were found as then validation accuracy didn't increase much but training accuracy did. This indicates that the model was trying to get overfitted.

Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
question_input (InputLayer)	(None, 100)	0	-
question_embedding (Embedding)	(None, 100, 300)	2,812,500	question_input[0…
question_lstm (LSTM)	(None, 100, 512)	1,665,024	question_embeddi…
question_dropout (Dropout)	(None, 100, 512)	0	question_lstm[0]…

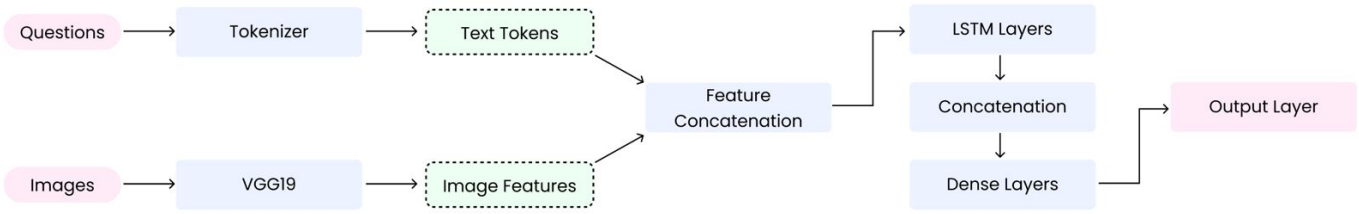
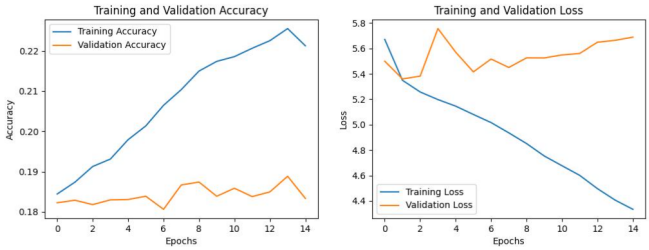


Fig. 1. Architecture of the Approach-1

image_input (InputLayer)	(None, 25088)	0	-
question_lstm2 (LSTM)	(None, 256)	787,456	question_dropout…
image_dense (Dense)	(None, 256)	6,422,784	image_input[0][0]
question_dropout2 (Dropout)	(None, 256)	0	question_lstm2[0…
image_dropout (Dropout)	(None, 256)	0	image_dense[0][0]
concatenated (Concatenate)	(None, 512)	0	question_dropout… image_dropout[0]…
dens_conc (Dense)	(None, 512)	262,656	concatenated[0][…
dens_conc2 (Dense)	(None, 512)	262,656	dens_conc[0][0]
output (Dense)	(None, 9134)	4,685,742	dens_conc2[0][0]

We had in-total 16,898,818 trainable parameters in the whole model.

Metric	Value
Val Accuracy	0.1834
F1 Score	0.1018
Precision	0.0725
Recall	0.1834
Time Taken for Training (TTT)	3252.28 sec



B. *Pipeline Explanation for Architecture - 1*

Image Input and Text Input: The raw images and textual questions are fed into the model as separate inputs.
VGG19 CNN and Tokenization: The images are processed by the VGG19 CNN to extract image features, while the textual questions are tokenized to convert them into sequences of tokens.
Image Feature Extraction and Tokenized Sequence: The outputs of the VGG19 CNN and the tokenization process are the image features and the tokenized sequence of the question, respectively.
Concatenation: The image features and the tokenized sequence are concatenated into a single feature vector.
LSTM Layers: The concatenated feature vector is then processed through LSTM layers to capture temporal dependencies in the textual data.
Concatenation: The outputs of the LSTM layers are concatenated with the image features.
Dense Layers: The concatenated features are passed through dense layers to extract relevant patterns and reduce dimensionality.
Output Layer: The final output layer predicts the answer based on the learned representations.
This pipeline diagram demonstrates the flow of information through the VGG19 + LSTM model, from input processing to final prediction generation.

VI. Approach-2

We have combined two well-known models to address the problem statement. The first model is BERT (Bidirectional Encoder Representations from Transformers), a groundbreaking model in natural language processing (NLP) developed by Google in 2018. BERT employs a bidirectional approach to read text, allowing it to understand the context of a word based

on all surrounding words, which significantly enhances its comprehension abilities. Built on the Transformer architecture, BERT uses self-attention mechanisms to handle long-range dependencies in text effectively. It is trained in two stages: pre-training on a large corpus to learn language patterns and fine-tuning on specific tasks to adapt to particular applications.

The second model we use is the Vision Transformer (ViT), an innovative architecture that applies the principles of Transformers, originally developed for NLP, to image recognition tasks. ViT divides an image into a sequence of fixed-size patches, which are then linearly embedded and processed similarly to tokens in a text sequence. By leveraging the self-attention mechanism of Transformers, ViT captures complex dependencies and relationships across different parts of an image, achieving state-of-the-art performance on several benchmark datasets. This approach has shown that Transformers can surpass traditional convolutional neural networks (CNNs) in image classification tasks, especially when trained on large-scale datasets, marking a significant advancement in computer vision.

Here's how we utilize these two models to solve the visual question answering problem:

A. Approach 2: ***ViT + BERT***

1) Feature Extraction:

- *Text Encoding*
  - We use a pretrained text encoder, BERT, which has been pre-trained on a large corpus of text data.
  - BERT transforms the input question text into a dense, contextualized representation, capturing not only the meaning of individual words but also their relationships within the context of the sentence or question.
  - Leveraging BERT, the model can understand the subtleties of language, including nuances, semantics, and syntactic structures, which are crucial for comprehending and accurately answering questions.
- *Image Encoding*
  - Similarly, we use a pretrained image encoder, Vision Transformer (ViT), which has been trained on extensive image datasets.
  - The image encoder processes the input image data and extracts high-level visual features representing various elements, patterns, and objects present in the image.
  - Vision Transformer breaks down the image into patches and transforms them into a compact, hierarchical representation that captures both local and global spatial relationships, enabling the model to perceive and interpret visual content effectively.

2) Feature Concatenation:

- Once the text and image features have been extracted, they are combined to create a unified representation that integrates information from both data types.
- This feature concatenation step is crucial for the model to understand the relationship between the question and the image and make informed predictions.
- By concatenating the text and image features into a single feature vector, the model effectively merges the complementary information provided by both data types, enhancing its ability to generate accurate and contextually relevant answers.

3) Classification:

- The concatenated feature vector undergoes further processing through a series of neural network layers, collectively referred to as the fusion module.
- The fusion module transforms the combined features into a more discriminative representation by applying non-linear transformations, such as linear projections followed by activation functions like ReLU (Rectified Linear Unit).
- The transformed feature vector is then passed through a classifier, typically a linear layer, which maps the features to the space of possible answers.
- The classifier generates logits, which are unnormalized scores representing the model's confidence in each possible answer.

B. Pipeline Explanation for Architecture - 2

Text Input and Image Input: The raw inputs to the model.
BERT Encoder: Processes the text input to generate contextualized word embeddings.
ViT Encoder: Processes the image input to generate high-level visual features.
Feature Concatenation: Combines the text and image features into a single vector.
Fusion Module:
Neural Network Layers: These layers apply linear transformations followed by ReLU activations to introduce non-linearity and enhance the discriminative power of the combined features.
Classification Layer: Maps the transformed features to possible answers and generates logits.
Predicted Answer: The final output of the model.
In the fusion module, each neural network layer typically consists of a linear transformation (fully connected layer) followed by a ReLU activation function. This process helps the model learn more complex representations of the concatenated features, improving its ability to generate accurate and contextually relevant answers.

VII. Testing & Evaluation Metrics

We evaluated the model using 10,000 image-question pairs from the dataset. Each pair was input into the model, which classified over a set of possible answers and selected the one with the highest logit value. The testing accuracy was approximately 48%.
No. of epochs : 5
Time taken : approx. 5 hours 45 minutes
Number of trainable parameters : 201,321,602

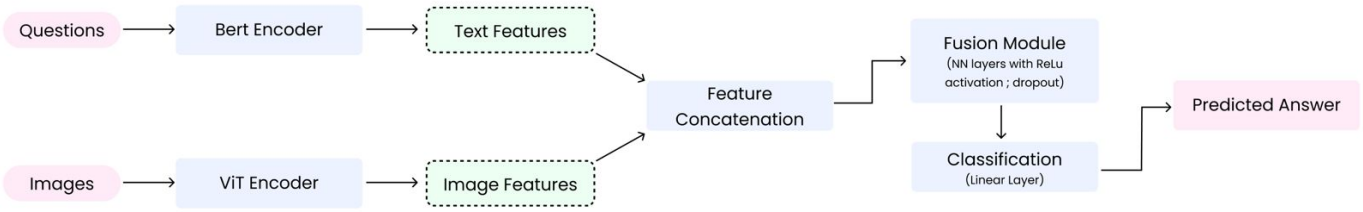


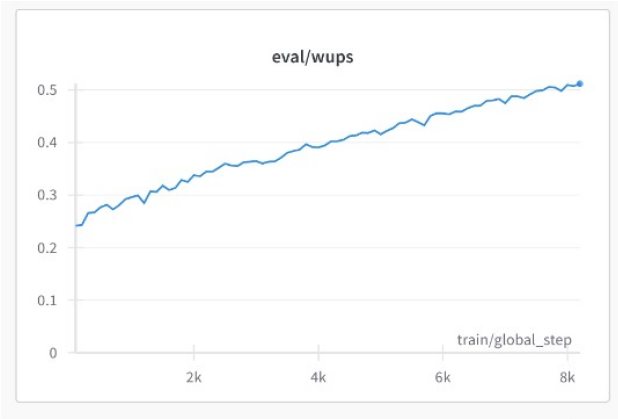
Fig. 2. Architecture of the Approach - 2

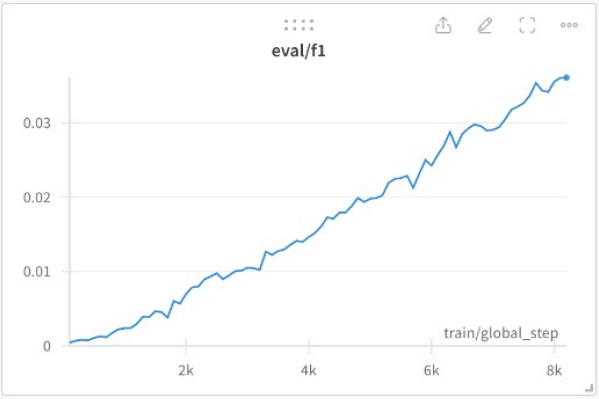
- **Wu & Palmer Similarity:** For multiple-choice tasks, simple accuracy works. For complex tasks like Visual Question Answering (VQA), exact matches are too rigid. Wu & Palmer (WUP) [2] similarity is a better alternative. It uses taxonomy to assess path length and depth, focusing on semantic similarity instead of exact matches. This is useful for single-word answers, providing a more nuanced evaluation of VQA models.
  WUP similarity: 0.5119907538065882
- **Recall:** Recall measures the proportion of actual positives that are correctly identified by a model.
  Recall: 0.04833948396336653
- **Precision:** Precision measures the proportion of positive identifications that were actually correct.
  Precision: 0.03649127152015741
- **F1 Score:** F1 score is the harmonic mean of precision and recall, providing a balanced evaluation of a model's performance.
  F1 Score: 0.03626461193780249

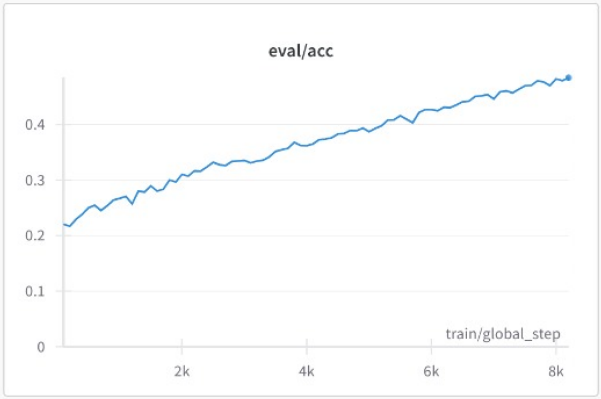
VIII. Usage of LoRA

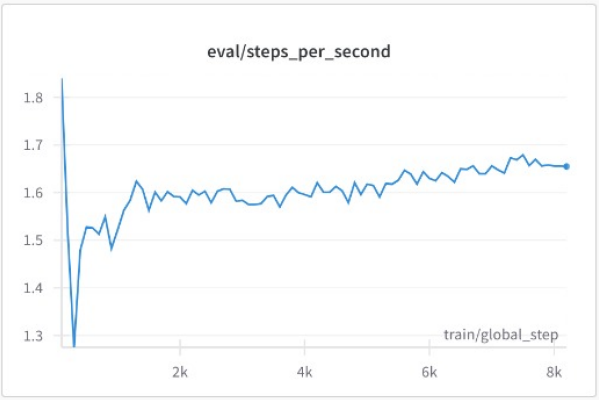
LoRA (Low-Rank Adaptation) is a technique for efficiently adapting pre-trained neural network models to specific tasks by introducing low-rank decomposition into weight updates. Instead of fine-tuning the entire model LoRA uses smaller matrices to adjust the weights, significantly reducing the number of trainable parameters. This approach makes the adaptation process faster, less computationally demanding and more memory efficient by maintaining the original pre-trained weights and only updating the smaller matrices.

We implemented LoRA in specific layers in both BERT and ViT models. The basically helped in decomposing the weight matrices in these layers into product of two smaller matrices. Other layers where LoRA is not used have their weights frozen and they are not updated using less computation.
BERT model with LoRA processes the input questions and ViT model with LoRA layers processes the input images. Extracted features from both BERT and ViT are combined. Co-Transformer layers further integrate these features, enhancing the interaction between textual and visual information. These combined features pass through dense layers to produce the final classification output.
LoRA with rank=8 significantly reduces the number of trainable parameters, making the model more efficient. This also helped in reduction of time of training. LoRA maintained performance of the model without LoRA and efficiently captured the patterns both in images and text.
Total time taken for model to get trained was around 30 mins for 10 epochs. We got validation Loss: 0.0172383, validation accuracy: 34.265%, validation precision: 0.1132, validation recall: 0.3321, validation F1: 0.1643



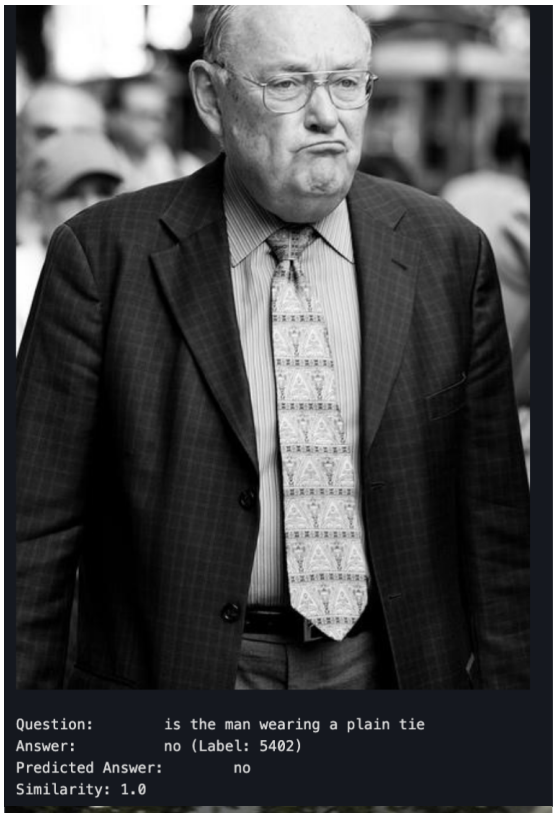






IX. Results

The following are some test results of our model:

- 

- 

'How many men are wearing hats?'

predicted answer: 1
Actual answer: 1

X. Conclusion

We noticed that Approach2, ViT+BERT, was better than Approach1, VGG19+LSTM. Training time for each epoch of VGG19+LSTM was around 5mins and for ViT+BERT without LoRA it was 1hr 09mins. ViT+BERT with LoRA took around 4mins.

Accuracy wise ViT+BERT without LoRA performed the best followed by ViT+BERT with LoRA and then VGG19+LSTM. Other evaluation metrics also followed the same order.

We tried changing the rank in LoRA implementation and figured out that amount of training time increased but didn't change the evaluation metrics significantly.

At the end, we can conclude that feature map created ViT is text feature created by BERT is much better than VGG19 approach which in turn helped in getting better evaluation metrics. LoRA was helpful in reducing the training time significantly but the evaluation metrics also dropped at the same time.

References

[1] D. Kanakamedala, T. Veeranki, R. Bitla, S. Vangalapudi and S. T, "Visual Question Answering Using Deep Learning," 2021 Innovations in Power and Advanced Computing Technologies (i-PACT), Kuala Lumpur, Malaysia, 2021, pp. 1-5, doi: 10.1109/i-PACT52855.2021.9696665

[2] Guessoum, Djamel Miraoui, Moeiz Tadj, Chakib. (2016). A modification of Wu and Palmer Semantic Similarity Measure.

[3] S. Chowdhury and B. Soni, "Visual Question Answering Optimized Framework using Mixed Precision Training," 2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1), Bangalore, India, 2023, pp. 1-6, doi: 10.1109/ICAIA57370.2023.10169318.