INTER IIT ML LITERATURE REVIEW

By, TEAM 1

1. VQA-RAD DATASET:

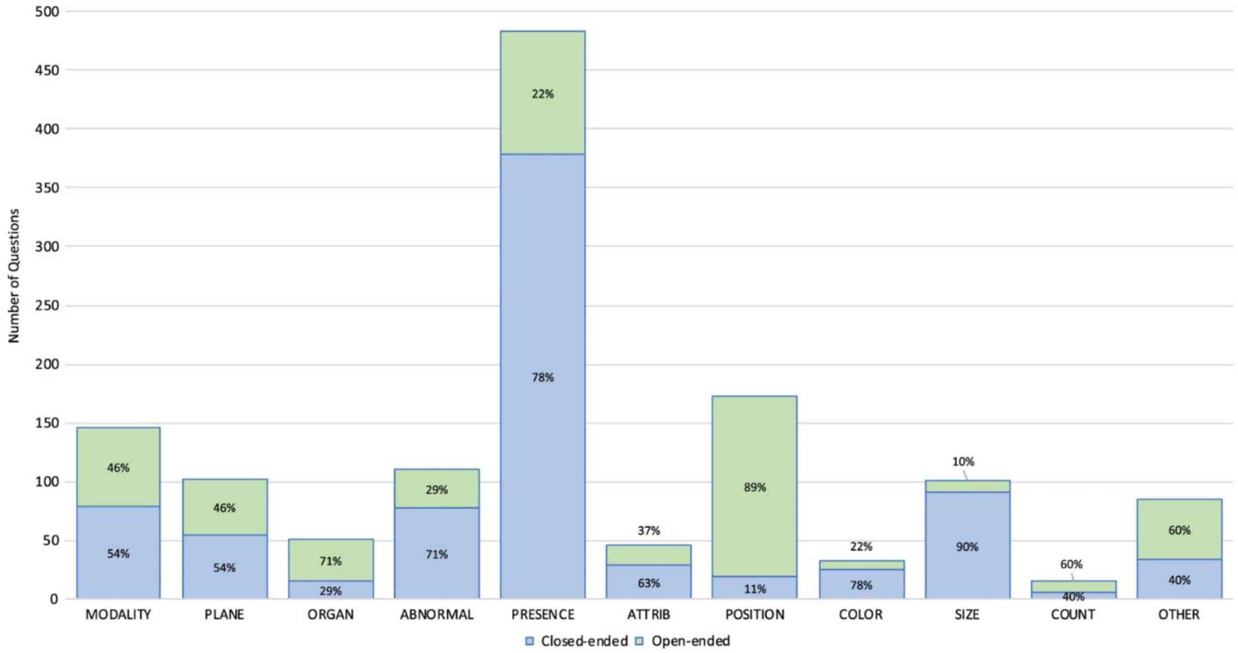
Resources: [A dataset of clinically generated visual questions and answers about radiology images | Scientific Data (nature.com)]

The VQA-RAD Dataset is the first manually constructed dataset where clinicians asked naturally occurring questions about radiology images and provided reference answers. It consists of 3515 question and answer pairs for a total of 315 images

SPECIFICATIONS

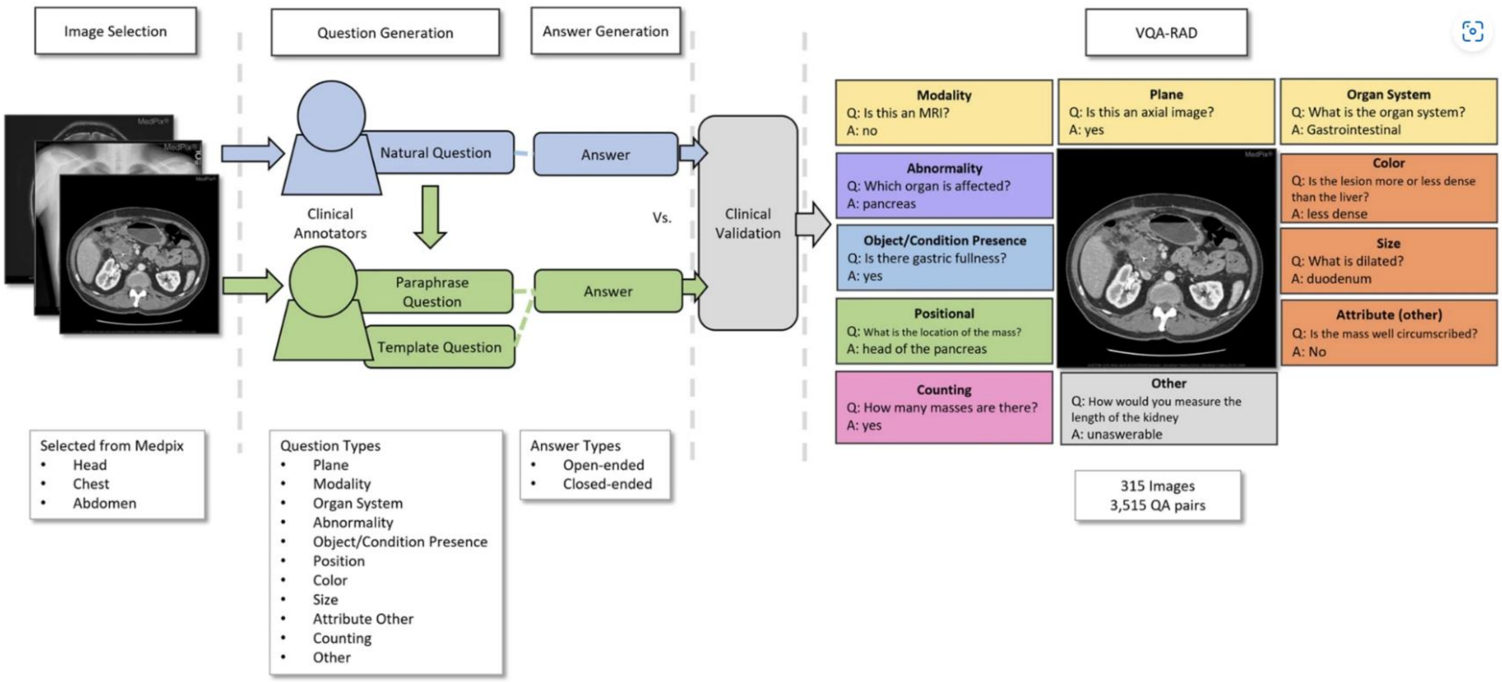
a. **Images:** The dataset comprises a substantial collection of radiological images of the Head, Chest and Abdomen, including X-rays, CT scans, and MRIs.

b. **Questions:** Questions are of different categories: Modality, Plane, Organ System, Abnormalities, Object/ Condition Presence, Color, Size, Attribute, Position, Counting and other



c. **Answer:** Answers are of 2 types – open-ended and close-ended.

The above results were found on EDA and on reading various research papers on the dataset.



VQA RAD BUILD

2. CURRENT CHALLENGES:

There are mainly 2 challenges in while dealing with VQA in radiology:

1. **Limited availability of data:** In the context of medical imaging, access to diverse and well-annotated datasets is essential for training robust machine learning models. Limited data restricts the capacity to capture the wide spectrum of medical conditions, anatomical variances, and imaging modalities
2. **Current VQA models are based on classification approach:** Most of the models treat answer generation as a classification task wherein they select an answer from the set of all possible answers. This works well with close-ended questions but provides very low accuracy with open ended questions. Consequently, they are only useful for limited use cases where a finite set of outcomes is provided beforehand.

Hence, we have devised an approach that carefully tackles both of the above challenges and hence, can define the problem statement as follows:

PROPOSED MODEL

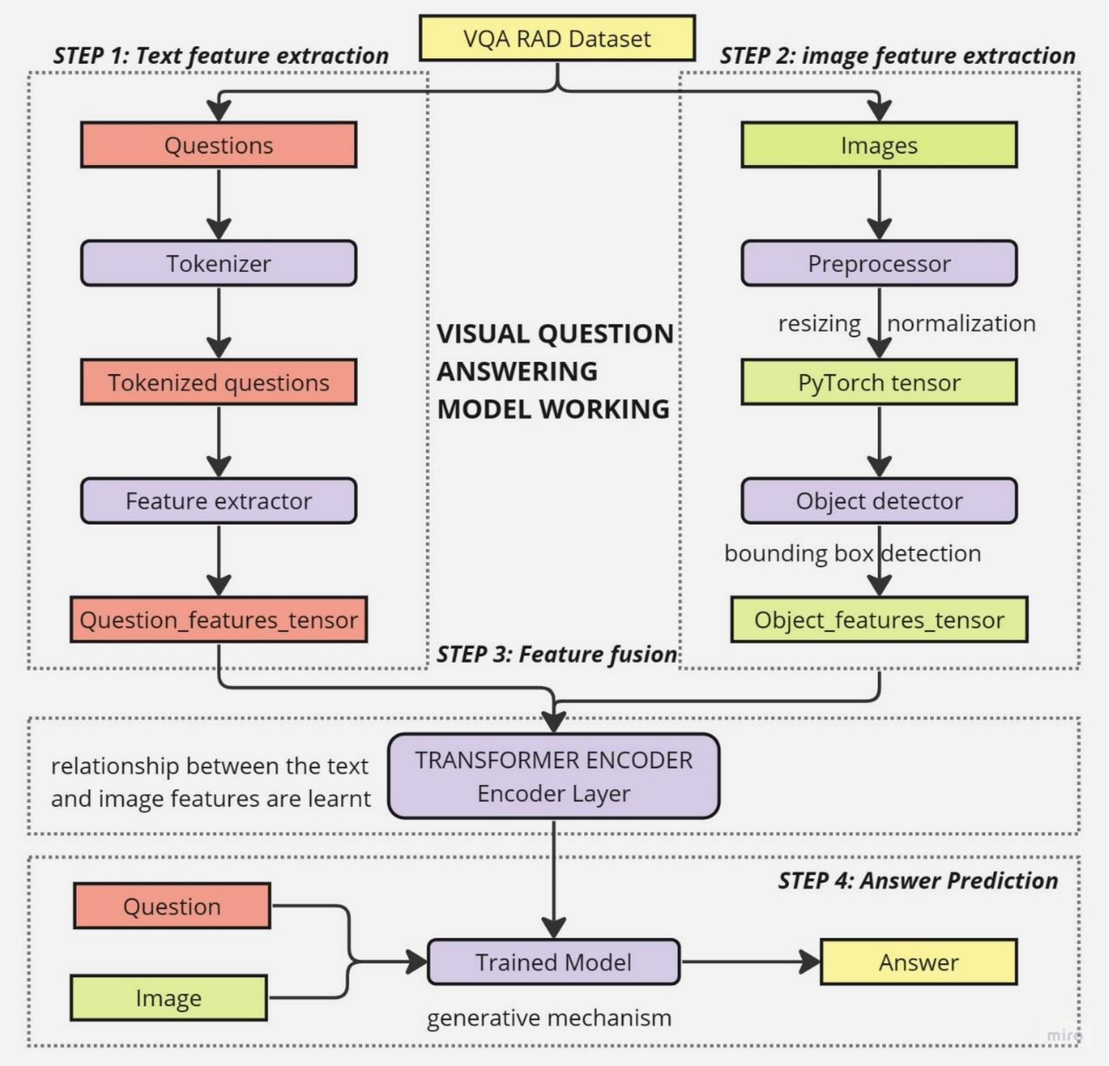
Resources:
https://drive.google.com/file/d/1VfLb4R3OK0mUjE6qBlOydMy47lulOyh2/view?usp=share_link

PROBLEM STATEMENT: Devise a generative multi-model model for visual question answering in radiology that accurately answers diverse questions

SOLUTION: An attention-based multimodal deep learning model for visual question answering in the medical domain.

Our model has 4 components:

- **a.** Question Answer feature Extraction – NLP Task
- **b.** Image feature Extraction – CV Task
- **c.** Feature fusion
- **d.** Answer Prediction



The 4 models we have used follow the 4 steps described above. Below is the detailed explanation of how each of the models implements the steps.

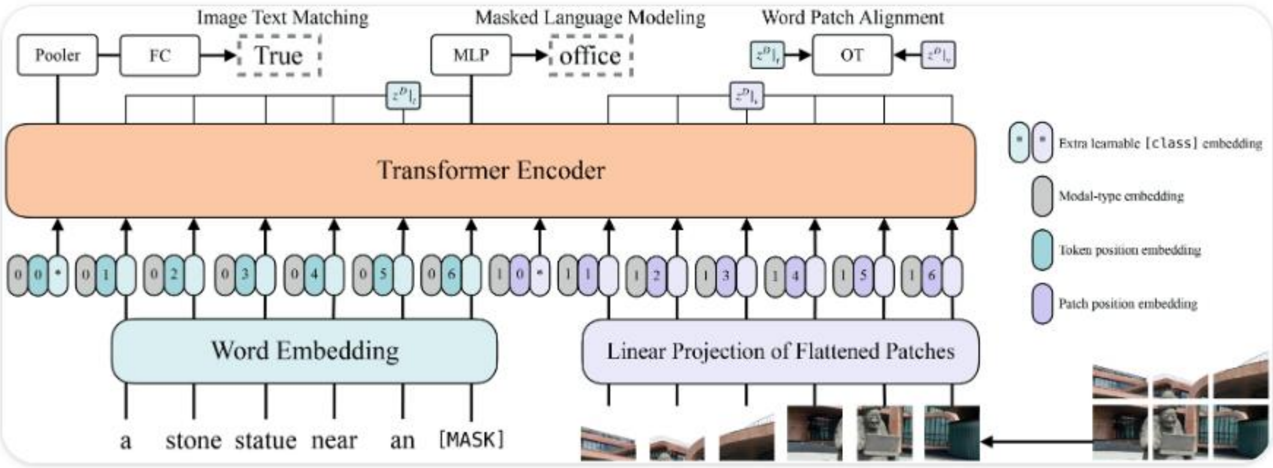
MODEL 1: Using ViLT Transformer

Resources: ViLT (huggingface.co)

The ViLT model was proposed in ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision by Wonjae Kim, Bokyung Son, Ildoo Kim. ViLT incorporates text embeddings into a Vision Transformer (ViT), allowing it to have a minimal design for Vision-and-Language Pre-training (VLP).

1. **Question feature extraction:** ViLT performs question feature extraction using its *Language Embedding Module*. The module *tokenizes* and encodes the input question into a sequence of tokens. These tokens are then passed through transformer layers to obtain *contextualized embeddings*. These embeddings capture the *semantic meaning* of the question in the context of other words in the sentence. The final output is a fixed-size feature vector representing the input question.
2. **Image feature extraction:** For image feature extraction, ViLT utilizes its *Visual Embedding Module*. This module treats images as sequences of tokens. Pretrained vision models, such as CNNs, are used to extract image features mainly through object detection. These features are then transformed into tokens. ViLT's Visual Embedding Module encodes these tokens into a sequence of embeddings, capturing visual information from the image.
3. **Feature fusion:** ViLT achieves feature fusion by combining the extracted question features and image features. The transformer layers in ViLT are designed to handle both modalities (text and images) seamlessly. The joint *multimodal embeddings* capture the relationships between words in the question and visual elements in the image. During training, the model learns to weigh the importance of different question words and image regions, leading to effective feature fusion.
4. **Answer Prediction:** The fused multimodal features are passed through additional transformer layers, enabling the model to learn complex patterns and relationships between the question and the image. The final transformer layer's output is used for answer prediction. Depending on the specific task (e.g., classification or regression), the output can be fed into an appropriate activation function (e.g., softmax for classification) to obtain the predicted answer.

WORKING OF ViLT TRANSFORMER



Specifications of ViLT Transformer:

1. **ViLT Feature Extractor:** preprocesses an image or batches of images
2. **ViLT Image Processor:** Constructs a ViLT image processor
3. **ViLT Processor:** Constructs a ViLT processor which wraps a BERT Tokenizer and ViLT Image processor into a single processor
4. **ViLTForQuestionAnswering:** This is a PyTorch torch.nn module which helps in the VQA task

Code: https://colab.research.google.com/drive/1zemYdzJSGKaO92JRsEcQqoymhDPzhInG?usp=sharing

MODEL 2: Using BLIP Transformer

The BLIP model was proposed in BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation by Junnan Li, Dongxu Li, Caiming Xiong, Steven Hoi. It specializes in VQA, Image text retrieval, image captioning. It is a more complex model as compared to ViLT transformer.

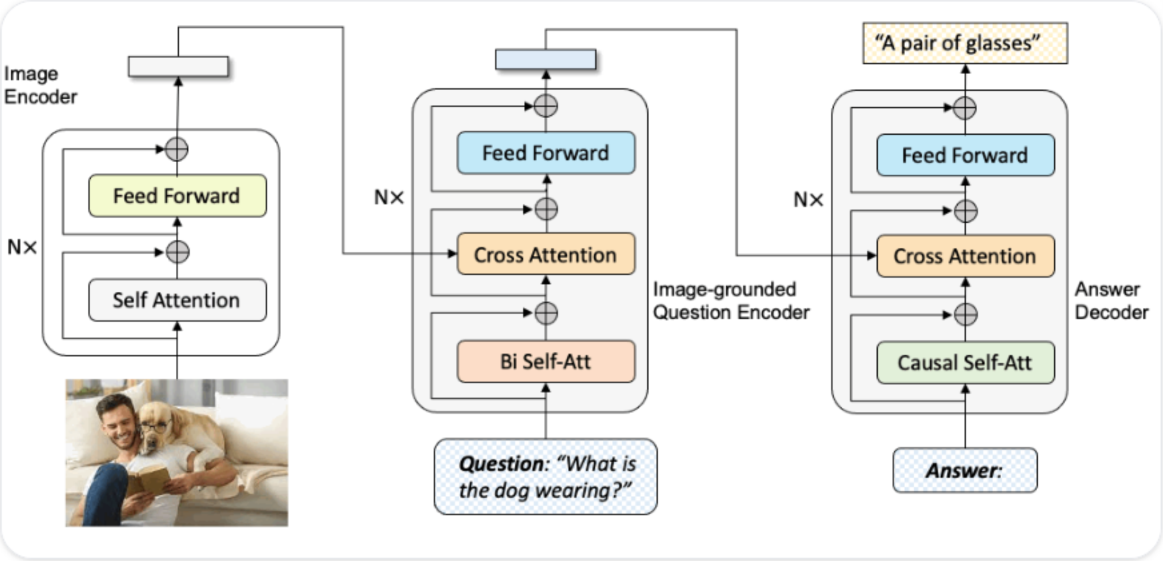
1. **Question Feature Extraction:** BLIP performs question feature extraction using *masked language modeling (MLM) techniques.* During pretraining, BLIP uses a masked language modeling objective, similar to BERT, where certain tokens in the input question are randomly masked, and the model is trained to predict these masked tokens based on the context of the remaining tokens. This process encourages the model to *learn deep contextualized representations* of words in the questions.
2. **Image Feature Extraction:** For image feature extraction, BLIP utilizes a *contrastive learning approach*. It learns to encode images into continuous representations by *maximizing the similarity between positive pairs* (different augmentations of the same image) and *minimizing the similarity between negative pairs* (augmentations of different images). This process encourages the model to capture relevant visual features that are consistent across different augmentations of the same image while being distinct from other images.
3. **Feature Fusion:** The fusion module can use techniques like *concatenation, element-wise addition, or more sophisticated fusion strategies* to create joint multimodal embeddings. The fusion step allows BLIP to capture cross-modal interactions between textual and visual information.
4. **Answer prediction:** The BLIP model must be fine-tuned for the VQA task where it is trained on a labeled dataset and the model learns to map the joint embeddings to the correct answers through various procedures like classification or regression.

Specifications of BLIP Transformer:

1. **BLIP Image Processor:** Constructs a BLIP image processor
2. **BLIP For Conditional Generation:** BLIP Model for image captioning. The model consists of a vision encoder and a text decoder. One can optionally pass input_ids to the model, which serve as a text prompt, to make the text decoder continue the prompt. Otherwise, the decoder starts generating text from the [BOS] (beginning-of-sequence) token.
3. **BLIPForQuestionAnswering:** The model consists of a vision encoder, a text encoder as well as a text decoder. The vision encoder will encode the input image, the text encoder will encode the

input question together with the encoding of the image, and the text decoder will output the answer to the question.

WORKING OF BLIP TRANSFORMER



<u>Code:</u>

All the above transformers performed all the 4 steps having in-built text and image encoders. We have also tried 2 more combinations in which different transformers perform the different tasks individually. The third model is inspired from the MCB (Multi-modal Compact Bilinear Pooling) and the fourth from SAN (Stacked attention networks) architecture.

MODEL 3: <u>RadBERT+ResNET152+LXMert</u>

RadBERT: RadBERT is a series of models trained with millions (more to come!) radiology reports, which achieves stronger medical language understanding performance than previous bio-medical domain models such BioBERT, Clinical-BERT, BLUE-BERT and BioMed-RoBERTa.

1. **Question Feature Extraction:** RadBERT performs the text feature extraction. It tokenizes the questions and answers. Once tokenized, they are fed into the transformer model and the model *obtains the outputs [features] for these tokens*. *The hidden features corresponding to the classification token* are then stored in another tensor which can then be used for further processing.

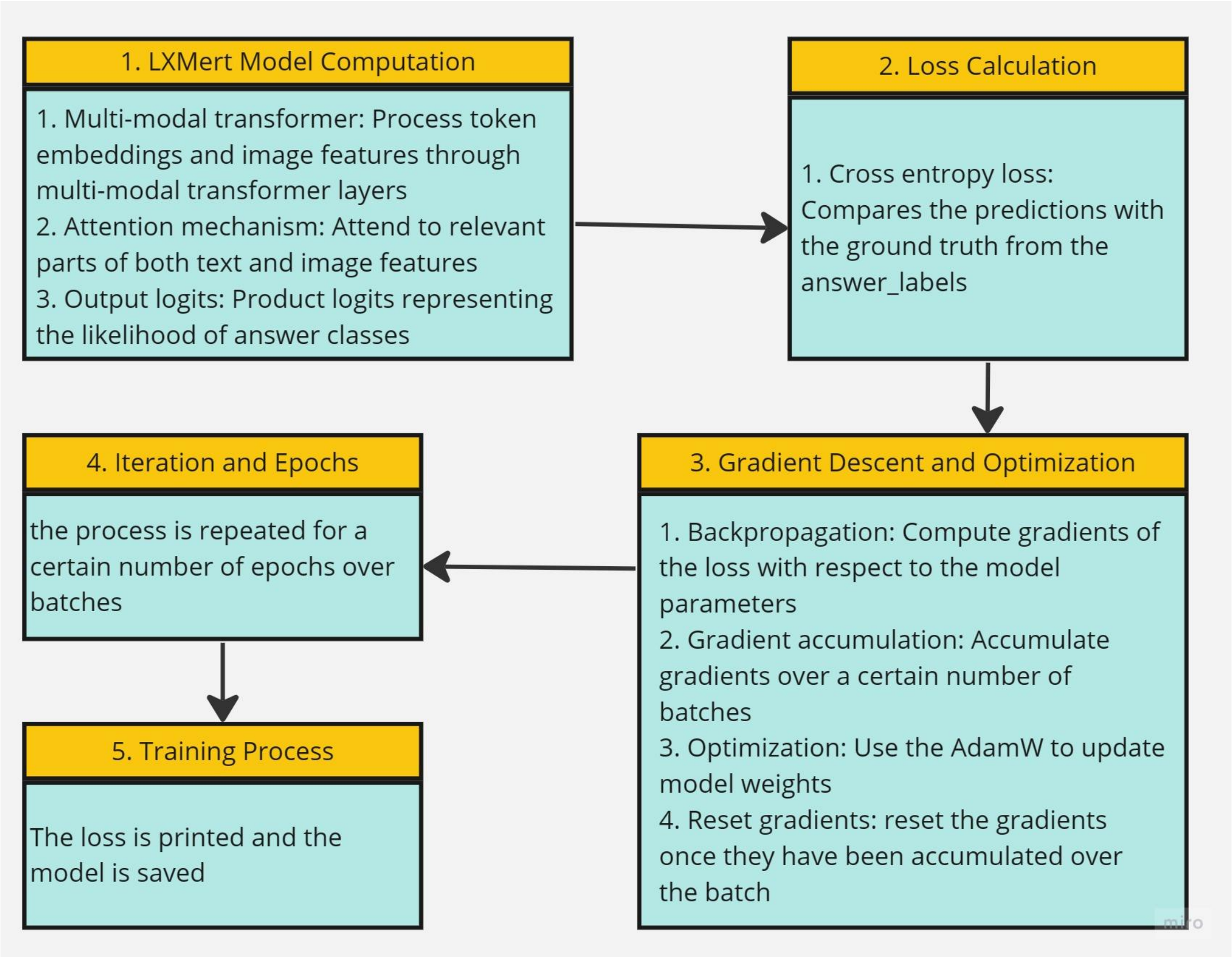
   An attention_mask tensor is also created from the question_features_tensor which holds *value "True" If for a corresponding non-zero value in the question_features_tensor and holds value "False" for a zero* in it. This tells the model which features to attend to and which to neglect.

2. **Image Feature Extraction:**

We have used a pre-trained **ResNET152** model to extract the features from the images. It does so in the following steps:

- The image is *converted to RGB* which is the standard input format for the ResNET model
- A sequence of image transformations are done to resize the images. The transformations include resizing the image to 256x256 pixels, cropping the center to 224x224 pixels (which is the input size expected by ResNet-152), converting the image to a PyTorch tensor, and normalizing the pixel values using the mean and standard deviation values specified in the transforms.Normalize function.
- Finally, the preprocessed tensor is passed through the ResNET model which gives the image_features_tensor
- The positional encoding tensor is used also created which is required by LXMert [being permutation-invariant] to understand the spatial relationship between pixels or patches. Images inherently possess a grid-like structure, and the spatial arrangement of pixels or patches carries vital information.
- The positional encoding tensor must be of the same size as the image_features_tensor.

WORKING OF LXMert



1. LXMert Model Computation

1. Multi-modal transformer: Process token embeddings and image features through multi-modal transformer layers
2. Attention mechanism: Attend to relevant parts of both text and image features
3. Output logits: Product logits representing the likelihood of answer classes

2. Loss Calculation

1. Cross entropy loss: Compares the predictions with the ground truth from the answer_labels

3. Gradient Descent and Optimization

1. Backpropagation: Compute gradients of the loss with respect to the model parameters
2. Gradient accumulation: Accumulate gradients over a certain number of batches
3. Optimization: Use the AdamW to update model weights
4. Reset gradients: reset the gradients once they have been accumulated over the batch

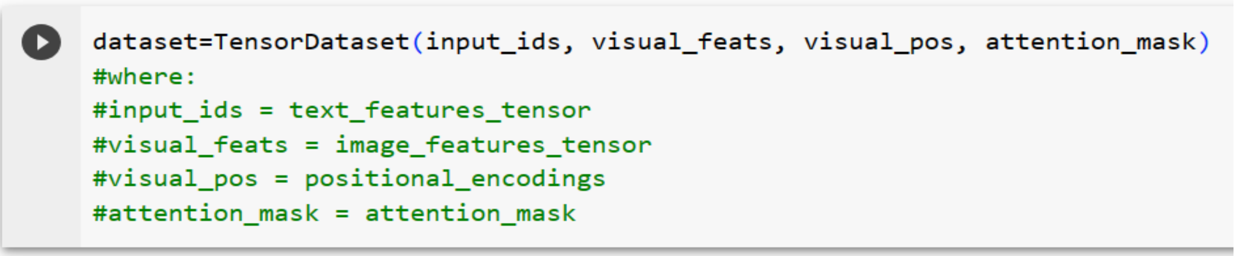
4. Iteration and Epochs

the process is repeated for a certain number of epochs over batches

5. Training Process

The loss is printed and the model is saved

3. **Feature Fusion:**
   Resources: LXMERT (huggingface.co)
   Fusion is performed using LXMert – a cross-modality pre-trained transformer.
   It is a series of bidirectional transformer encoders (one for the vision modality, one for the language modality, and then one to fuse both modalities) pretrained using a combination of masked language modeling, visual-language text alignment, ROI-feature regression, masked visual-attribute modeling, masked visual-object modeling, and visual-question answering objectives.
   LXMert takes input of tensors in the form of a TensorDataset.



```
dataset=TensorDataset(input_ids, visual_feats, visual_pos, attention_mask)
#where:
#input_ids = text_features_tensor
#visual_feats = image_features_tensor
#visual_pos = positional_encodings
#attention_mask = attention_mask
```

4. **Answer Prediction:** Once the TensorDataset has been passed onto the LXMert model, the training loop is run and the model learns various cross-modality features. Once this is done, it is then evaluated where it is fed a set of questions and images and it predicts the answer.

Code: https://colab.research.google.com/drive/1Qu95SF4iBdM-SEBR404rrRiafAwzNPzV?usp=sharing

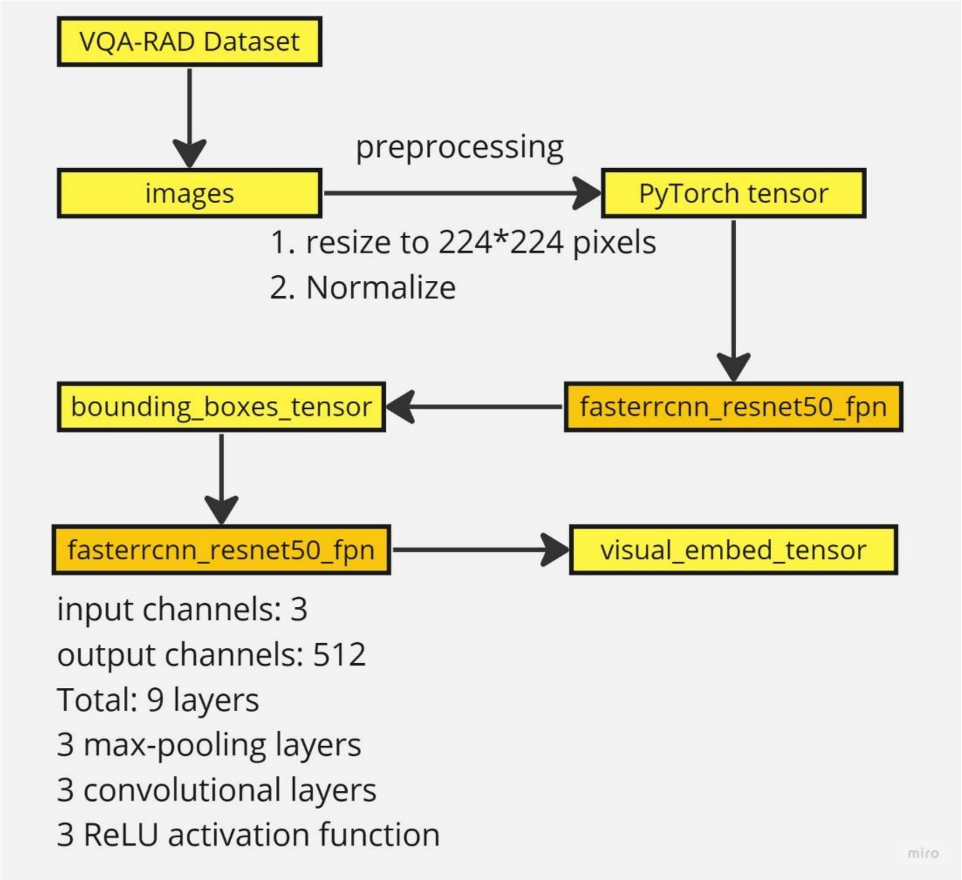
MODEL 3: BERT+Fasterrcnn_resnet50_fpn+VisualBERT

1. **Question Feature Extraction:** BERT transformer is used to tokenize the question and answers and then converted to PyTorch tensors
2. **Image Feature Extraction:**
3. For this we use a fasterrcnn_resnet50_fpn, which is a special model for object detection and it extracts the bounding boxes from the images which are then stored in a PyTorch tensor.
   Bounding boxes: Boundign boxes are rectangular frames used for object detection. They are defined by coordinates (x1, y1, x2, y2) representing the top-left and bottom-right corners of the box.
   ***objects** are distinct and recognizable entities in an images.
   Once the object_features_tensor is obtained, we pass it through a CNN to get the visual_embeds_tensor. The visual_embeds tensor stores all the information learnt by the CNN.

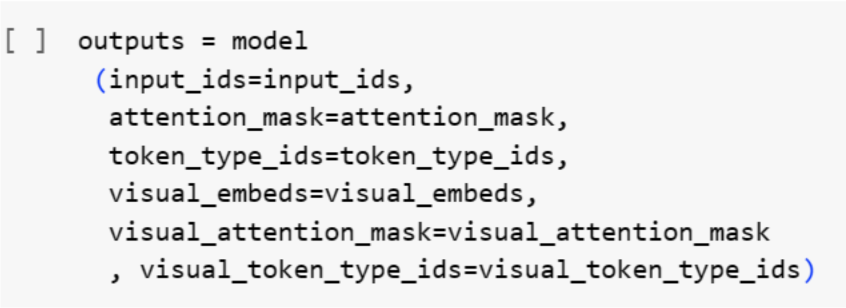
4. **Feature Fusion**
   Resources: VisualBERT (huggingface.co)
   We use **VisualBERT** to perform feature fusion
   VisualBERT consists of a stack of Transformer layers that implicitly align elements of an input text and regions.
   It takes the following as input:

Input_ids
Attention_mask
Token_type_ids, for the text feature processing, and:
Visual_embeds
Visual_attention_mask
Visual_token_type_ids, for the image_feature processing.

IMAGE PROCESSING BY FATERRCC_RESNET50_FPN



The answer prediction step remains the same. We don’t have to write the code for the entire training process as it is in-built. We can directly store the outputs of the model using the below line of code:



```
[ ]  outputs = model
      (input_ids=input_ids,
       attention_mask=attention_mask,
       token_type_ids=token_type_ids,
       visual_embeds=visual_embeds,
       visual_attention_mask=visual_attention_mask
       , visual_token_type_ids=visual_token_type_ids)
```

The features in the images can be more accurately determined by using Detectron2 which is our 3rd model. However, the model is too complex to be run on google colab.

We also suggest 2 better approaches, however, we couldn’t implement them due to lack of compatibility of our personal computers. They can be applied on computers which are compatible to run complex ML models.

Code: https://colab.research.google.com/drive/1UKHllQjsXz6cKvpD6mxF-bdlArORxLQ0?usp=sharing

OTHER APPROACHES

1. Training on the PMC_VQA Dataset:

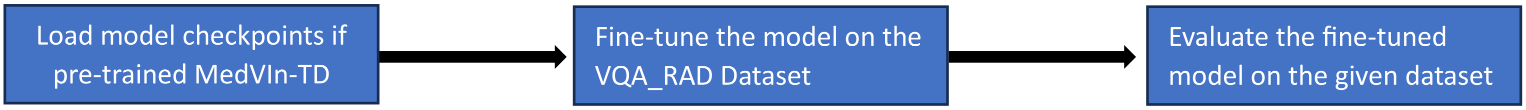
Resources: PMC-VQA Dataset | Papers With Code
2305.10415.pdf (arxiv.org)

In order to effectively train the generative-based MedVQA models, our *study reveals that existing datasets are limited in size*, making them insufficient for training high-performing models. To overcome this challenge, we leverage well-established medical visual-language datasets and initiate a scalable, automatic pipeline for constructing a new large-scale medical visual question�answering dataset. This new dataset, termed as **PMC-VQA**, contains 227k VQA pairs of 149k images, covering various modalities or diseases, surpassing existing datasets in terms of both amount and diversity.

Approach: Train the VQA model on the PMC_VQA dataset and then finetune it on the VQA_RAD dataset

Pre-trained models on PMC_VQA: 2 models namely the MedVInT-TE and the MedVInT-TD have already been trained on this dataset. Both of these have a Textual encoder, visual encoder and ta multi-modal decoder.

We will be using the MedVIn-TD pre-trained model as it has better accuracy on the VQA-RAD dataset.


Load model checkpoints if pre-trained MedVIn-TD
Fine-tune the model on the VQA_RAD Dataset
Evaluate the fine-tuned model on the given dataset

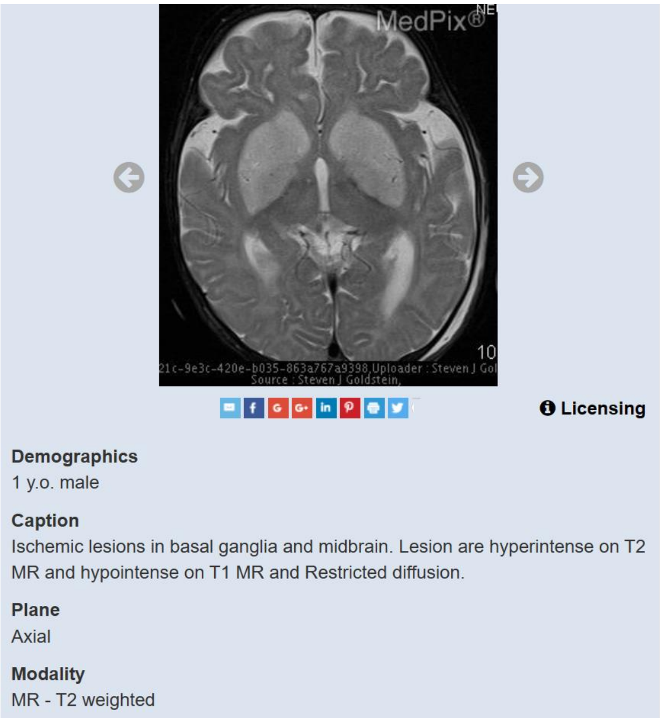
The MedVIn-TD pre-trained model gives an accuracy of 73.7 % on Open-ended questions, 86.8 % on close ended questions and hence, an overall accuracy of 81.6 %.

2. Q&A Pair Generative Model:

In this we propose the use of the information given with each image. The approach is explained in detail with the help of an example.

Consider the image: synpic47737 which has an associated link:
https://medpix.nlm.nih.gov/case?id=e7ee4900-44f3-4ea5-92c5-8d599c9c4b30

The link has the following details:


MedPix®
10
21c-9e3c-420e-b035-863a767a9398,Uploader : Steven J Gol
Source : Steven J Goldstein,

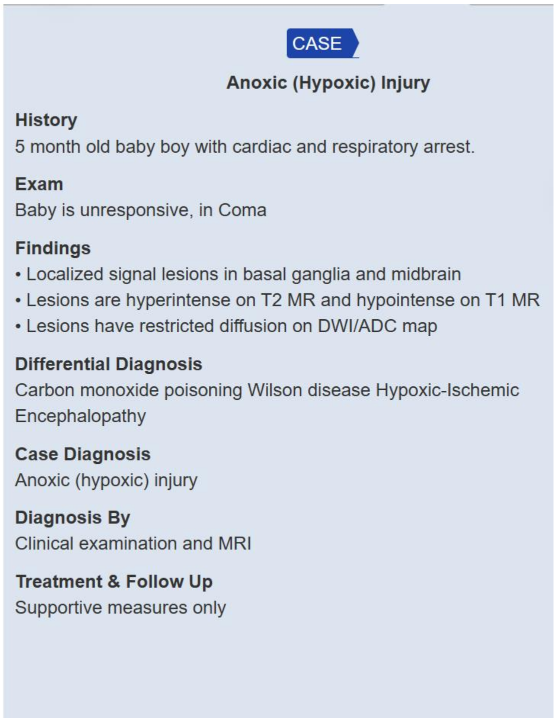
ℹ Licensing

Demographics
1 y.o. male

Caption
Ischemic lesions in basal ganglia and midbrain. Lesion are hyperintense on T2 MR and hypointense on T1 MR and Restricted diffusion.

Plane
Axial

Modality
MR - T2 weighted



CASE

Anoxic (Hypoxic) Injury

History
5 month old baby boy with cardiac and respiratory arrest.

Exam
Baby is unresponsive, in Coma

Findings
- Localized signal lesions in basal ganglia and midbrain
- Lesions are hyperintense on T2 MR and hypointense on T1 MR
- Lesions have restricted diffusion on DWI/ADC map

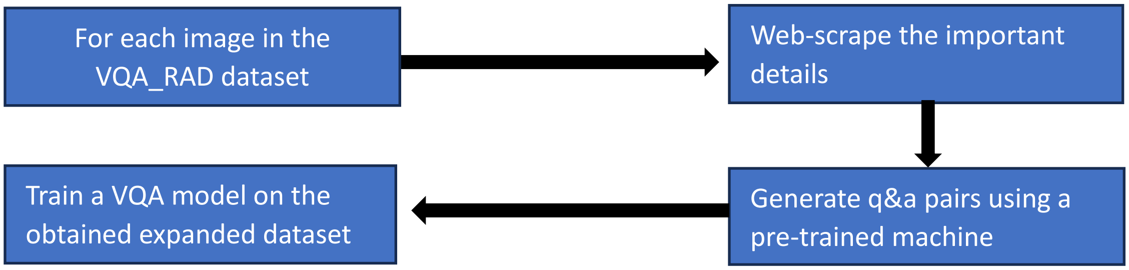
Differential Diagnosis
Carbon monoxide poisoning Wilson disease Hypoxic-Ischemic Encephalopathy

Case Diagnosis
Anoxic (hypoxic) injury

Diagnosis By
Clinical examination and MRI

Treatment & Follow Up
Supportive measures only

We can use these details to form more question and answer pairs for the image to make our dataset more diverse and capture the details.



For each image in the VQA_RAD dataset

Web-scrape the important details

Generate q&a pairs using a pre-trained machine

Train a VQA model on the obtained expanded dataset

New dataset created after webscraping: https://drive.google.com/file/d/1C2doZJfTLLsxnC9w7bDCYE-K-kmSosra/view?usp=sharing

As a result, we get a model which has much more accuracy than if trained on the limited q&a pairs provided.