

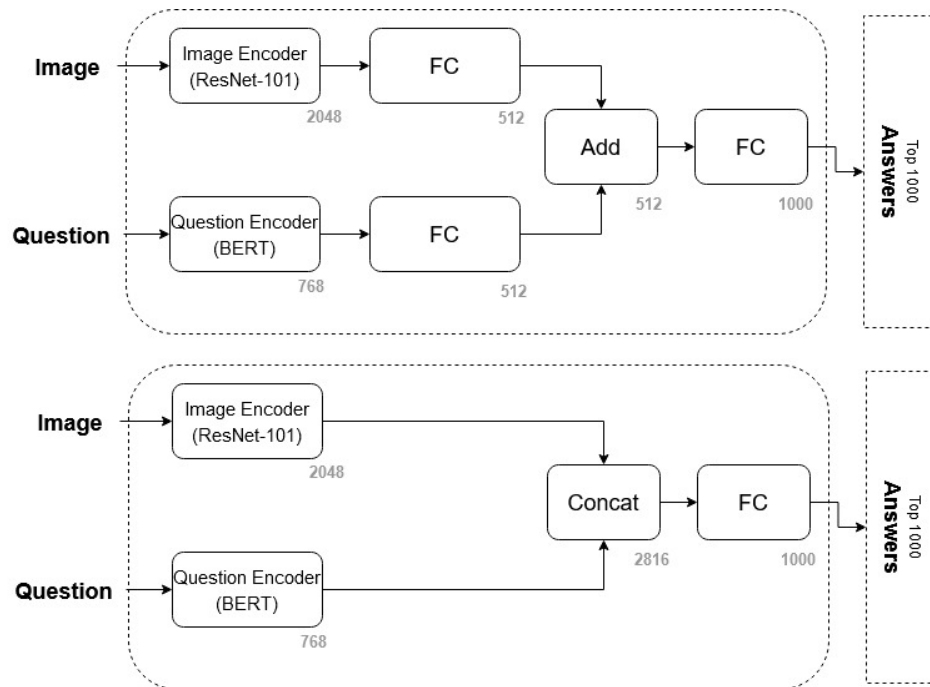
Brief Problem Statement

Visual Question Answering (VQA) is the process of creating an intelligent model that can identify an answer given simply an image and a question. [1] In this assignment we are supposed to work with the same input: an image and a question. The dataset that is provided is the Balanced Real Images set from the VQA Challenge site. [2] The training set contains about 80K images from COCO along with 443K questions and answers. The validation set contains 40K images and 215K questions with answers.

Description of Approach

This problem is a multi-modal problem where we need to somehow combine features learnt from both the images, and the text to classify into different answers. For that, we will also need feature extractors. Keeping this in mind, I have tried slightly different approaches treating this as a multi-class classification problem, detailed below.

- *Image Feature Extraction:* I use the provided ResNet-101 features
- *Text Feature Extraction:* I use the pre-trained BERT variant *bert-base-uncased* from HuggingFace for this along with the corresponding pre-trained tokenizer, which itself is based on WordPiece. [3]
- *Combining Image and Text Features:* I employ two ways here:
 - For the first one, I bring the textual and image features to the same number of dimensions using FC layers. That is, I reduce textual feature dimension size from 768 to 512, and image feature dimension size from 2048 to 512. Next, I add these two features and use one final FC layer to map to one of the top 1000 answers.
 - For the second one, I just concatenate the 2048 dimension image feature with the 768 dimension text feature to get the final feature, which I map to one of the top 1000 answers.
- *Training the model:* As mentioned above, and in the baseline method the space for the answers is very large so I focus only on the top 1000 occurring answers while prediction. To that end, I remove examples from the training set that contain answers not belonging to these top 1000. Using the remaining examples, the model is trained using either of the two architectures described above using Cross Entropy Loss for 50 epochs. The optimizer used is Adam with a learning rate of 0.001



Accuracy






As mentioned above, I tested two slightly different architectures for the model. These are the accuracies obtained:


- **Concat:** (50 epochs)
- **Add:** 48.07% (50 epochs)
- **2-LSTM:** 45.04% (50 epochs)

Examples

All of the below examples are output by the *Add* model. And as we can see, for the first examples, the answers are correct but for the last two the answers are wrong. Nonetheless, the model is able to figure out there is a gap between the wall and the roof and sees it as being “broken” in the last one and in the second to last, it is able to figure out that it is being asked to count something and outputs a number

Image ID	Question	Ground Truth	Prediction	Image
----------	----------	--------------	------------	-------

262162	How many beds?	1	1	
262162	Is that a folding chair?	no	no	
262197	Is there a tree in front of the building?	yes	yes	
262197	How many buildings?	2	1	
240301	Is it daylight in this picture?	yes	yes	

240301	Why is there a gap between the roof and wall?	For air	broken	
--------	---	---------	--------	---

Ablation

I decide to check what the difference in performance is if I just use the image features or just the text features versus using both.

	Both	Image	Text
Add	48.07%	5.16%	36.49%
Concat	42.73%	-	-
2-LSTM	45.04%	13.43%	34.65%

Reflection

While initially trying to build this system, I came across a very weird and specific problem which is that in Pytorch, the Cross Entropy Loss function takes the input in a particular format. This loss function does not work with one-hot encoded targets, and performs better mathematically if the predictions passed are logits and not probabilities (after Softmax).

I had done exactly this: Passed probabilities as predictions, and one-hot encoded targets. Due to this, the training loop was breaking after the first update, and predicting only one class for each example. I fixed this by following the documentation as described above

Next Steps

Some next steps I would and did try are listed below:

- Implemented the provided baseline of GloVe embeddings + LSTM with similar but slightly worse performance as listed in the table above
- I would also try other embedding methods such as word2vec, and BoW models.

References

- [1] Antol, Stanislaw, et al. "VQA: Visual Question Answering", Proceedings of the IEEE International Conference on Computer Vision, 2015.
- [2] Goyal, Yash et al. "Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering", CVPR, 2017.
- [3] "bert-base-uncased", <https://huggingface.co/bert-base-uncased>