Vision-Language Modeling for Real-World Accessibility: A VQA System for VizWiz

Suman Mandava, Venkata Ramana Murthy Polisetty, Sharon Dev Saseendran

1. Introduction:

Visual Question Answering, or VQA for short, is one of those exciting tasks in AI that brings together computer vision and natural language understanding. The idea is pretty simple on the surface: you give a system an image and a question about that image, and it gives you an answer. But under the hood, it's a pretty challenging problem. The model needs to "see" what's in the image, "understand" the question, and then "reason" about both to come up with the right answer.

Think of a scenario like this: you show a picture of a cat sitting on a couch and ask, *"What is the animal doing?"* A good VQA system should figure out that there's a cat and that it's probably resting or sitting. Seems easy for us humans, but not so much for machines.

Traditionally, VQA systems required building complex deep learning models that needed to be trained from scratch on massive datasets. This often took a lot of time, computation, and data engineering. But now, with the rise of large pre-trained models like OpenAI's CLIP, things have become a lot more flexible and efficient.

CLIP is a powerful model trained on hundreds of millions of image–text pairs from the internet. What makes it special is that it learns to understand both images and text in the same space. So, instead of training a new model from scratch, we can use CLIP to get smart "embeddings" (a kind of vector representation) for both the image and the question. Then, we just build a simple classifier that looks at these embeddings and predicts the answer.

In our project, we did exactly that. We used CLIP to process the image and the question, combined their embeddings, and trained a lightweight classifier on top to pick the right answer. This approach is efficient, avoids long training times, and takes advantage of the powerful understanding that CLIP already has.

In the rest of the report, we'll walk through how we built this system, the choices we made, how we trained and evaluated it, and what results we got.

2. Dataset Overview – VizWiz VQA:

For this project, we're using the VizWiz Visual Question Answering (VQA) dataset, one of the most unique and realistic datasets out there for VQA tasks.

What makes VizWiz stand out is that it's not just a collection of stock images with random questions. Instead, it's based on real images taken by blind or visually impaired people using mobile phones, along with spoken questions they asked about those images. These questions were later transcribed and answered by crowd workers.

So, this dataset introduces a ton of real-world challenges, like:
- Blurry or poorly framed images, because they're taken by people who can't see them
- Incomplete or unclear questions, since they're spoken out loud in the moment
- Unanswerable questions, like asking "What color is the jacket?" when the jacket is barely visible

2.1 Key Stats:

1) Around 31,000 image-question pairs
2) Comes split into train, validation, and test sets
3) Each question is paired with 10 human-provided answers
4) Images are natural and unfiltered, taken in everyday life situations

2.2 Why it's a great fit:

- It pushes models to be robust to noisy and imperfect input—just like real life.
- It covers a wide range of questions: yes/no, counting, descriptions, object identification, etc.
- It encourages models to learn when not to answer (which is super important in safety-critical applications).

All in all, VizWiz is an excellent benchmark if you want to test how well a VQA model holds up outside of the lab and in the real world.

3. Preprocessing

Before we can train our Visual Question Answering model, we need to properly understand and prepare our data. This step is all about getting the dataset into a format that's clean, structured, and ready to be used with our CLIP model. Here's what we did in our preprocessing pipeline:

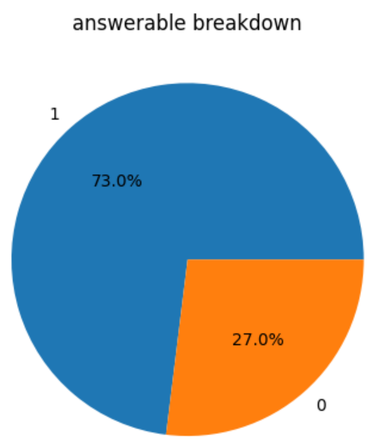
3.1. Reading and Exploring the Dataset

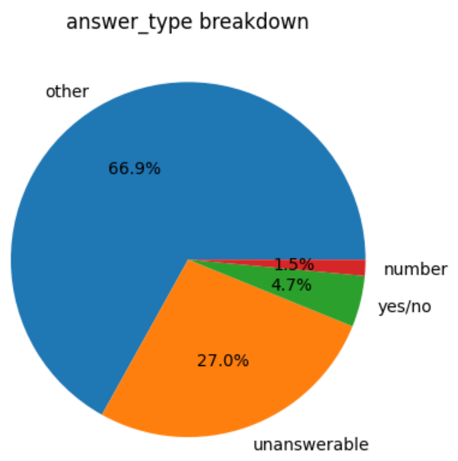
We started by loading the VizWiz dataset from its JSON annotations. Each entry in the dataset contains:
- The image filename
- A natural language question
- A list of answers from multiple annotators
- The answer type (e.g., "yes/no", "number", "other")
- A flag for whether the question is answerable

We only kept the columns we needed for training and analysis.
To better understand our data, we used some quick visualizations:

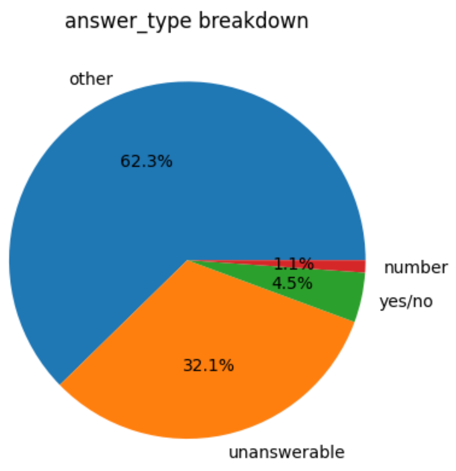
- **Pie charts** to see the distribution of different answer types and answerability

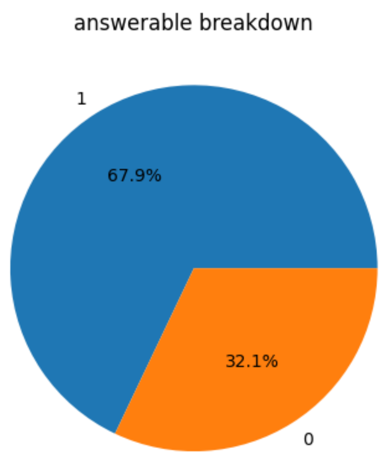
1) **Training:**



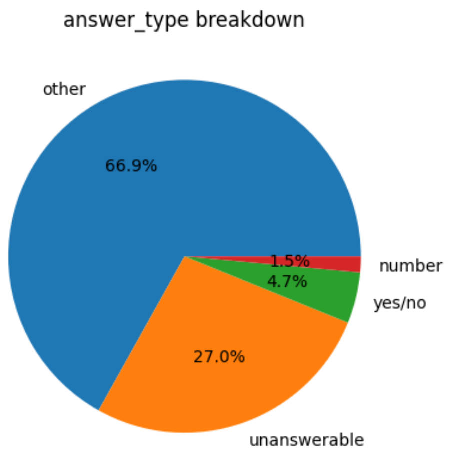


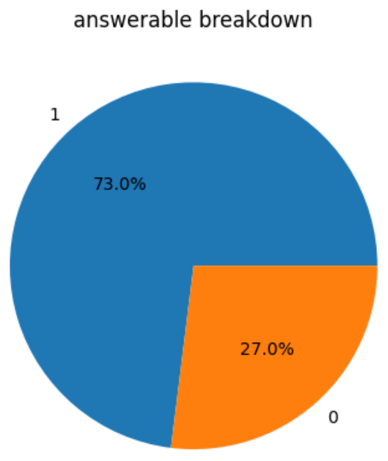
2) **Validation:**





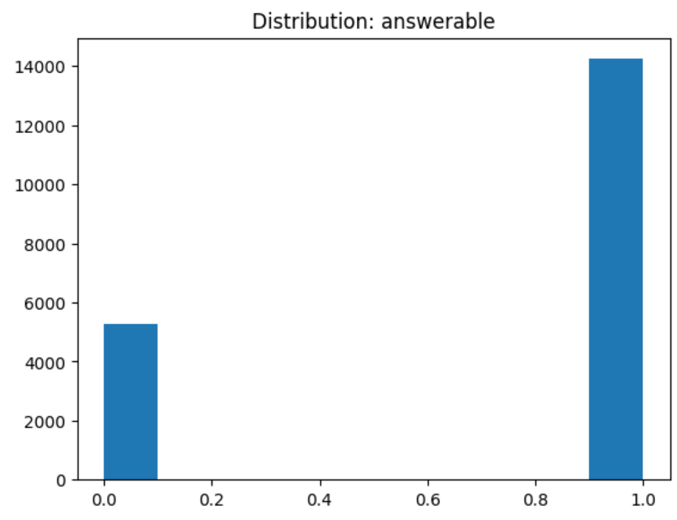
3) **Testing:**



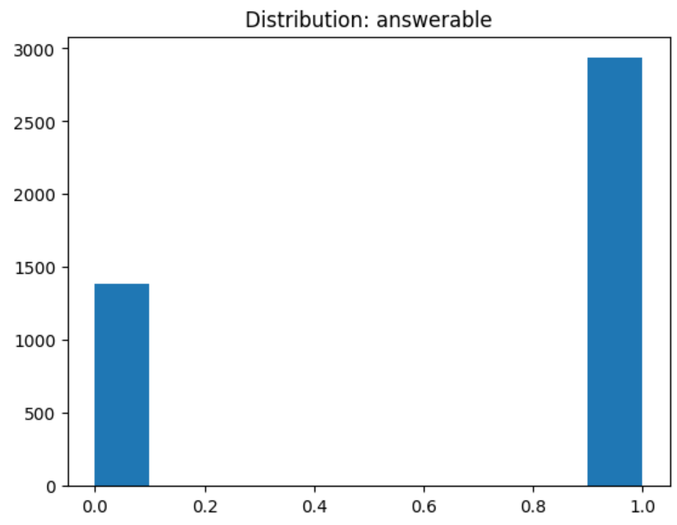


- **Histograms** to understand the balance of answerable vs. unanswerable questions

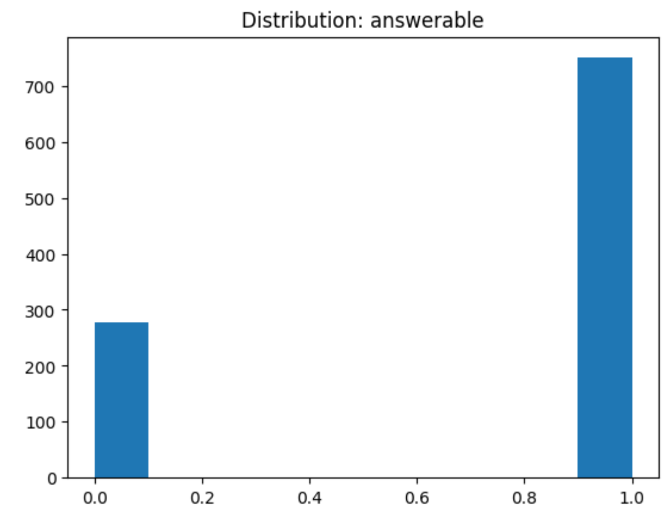
1) **Training :**



2) **Validation:**



3) **Testing:**



3.2. Splitting the Dataset

We split our dataset into **training and testing sets** using a stratified approach. This means the split maintains the original proportions of answer types and answerability status, which is important for balanced evaluation.

We used a small test size (like 5%) to keep most data for training.

3.3. Processing Questions (Text Features)

For each question, we used OpenAI's CLIP model to generate a text embedding:

- Tokenized the question using clip.tokenize()
- Encoded it with CLIP's encode_text() function
- Flattened the output into a single feature vector

These embeddings capture the semantic meaning of each question, making them useful for learning relationships between the question and the image.

3.4. Processing Images (Image Features)

For the visual side, we loaded each image using the provided path and:

- Pre-processed it with CLIP's official transform pipeline (resize, normalize, etc.)
- Passed it through clip.encode_image() to get a high-dimensional image embedding
- Flattened the image features for downstream use

We also added exception handling to skip over any broken or unreadable images without crashing the entire pipeline.

3.5. Counting Unique Answers

We scanned the dataset to count the number of unique answers across all questions. This helped us decide how many output classes our classifier needs. Since each question has multiple human answers, we also had to decide how to handle variations like synonyms or typos later (e.g., in label cleaning or majority voting).

4.**Model Architecture:**

We are following the architecture mentioned in , "Less Is More: CLIP-based Simple and Efficient VQA". The idea is to build a simple, lightweight, and effective model that takes full advantage of pre-trained representations from CLIP, while minimizing additional parameters and complexity.

4.1 Key Design Highlights:

- We freeze the CLIP model, using it purely as a feature extractor for both image and text inputs.

- We fuse the encoded image and question vectors using simple concatenation.
- We use fully connected layers to classify the final output into a fixed set of possible answers.
- The model jointly predicts:

  - The answer
  - The answer type (e.g., yes/no, number, other)
  - Whether the question is answerable

4.2 Architecture Overview

Here's how the model is structured, layer by layer:

Feature Extraction with CLIP

- Image: Encoded using CLIP's ViT-B/32 visual encoder, which processes the image as 32×32 patches and outputs a visual embedding.
- Text: Encoded using CLIP's transformer-based text encoder, which tokenizes and embeds the question.
- The resulting image and text embeddings are flattened and concatenated to form a unified multimodal feature vector.

Fusion + Classification

- The fused features are passed through two Linear Layers, each consisting of:

  1. Layer Normalization

  2. Dropout (p = 0.5)

  3. Fully Connected Layer (size = 512)

Multiple Heads for Different Outputs

- Answer Classifier → Predicts the final answer from a fixed answer vocabulary
- Answer Type Classifier → Predicts one of four predefined answer types (e.g., yes/no, number, etc.)
- Answer Mask Generator → Learns a dynamic mask over answer classes based on the predicted answer type
- Answerability Predictor → Outputs a binary label indicating whether the question is answerable from the image

Loss Functions

We trained the model using the Adam optimizer with a learning rate of 5e-4 and no weight decay. To train the model across its diverse outputs, a multi-loss approach is used:

- Cross Entropy Loss for:
  - Final answer classification
  - Answer type classification
- Binary Cross Entropy Loss (BCE Loss) for:
  - Predicting question answerability

This multi-head, multi-loss structure allows the model to reason better across different types of questions, ignore unanswerable queries, and generalize effectively crucial for challenging datasets like VizWiz.

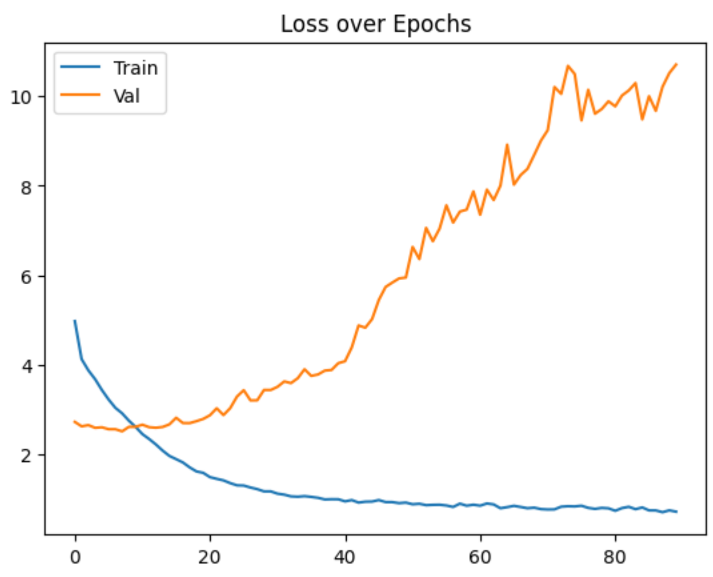
5. Training:

We trained our baseline VQA model for 90 epochs using the ViT-B/32 backbone with a data subset to ensure efficient local experimentation. Each epoch completed in approximately 1 minute, allowing us to iterate and analyze trends quickly. The best model checkpoint was selected from **epoch 85** based on validation accuracy and VizWiz evaluation metrics.

Due to resource constraints, we trained on a representative subset of the dataset. This setup was chosen for faster experimentation and to validate the model pipeline. Future iterations will extend training to the full dataset using the same framework.

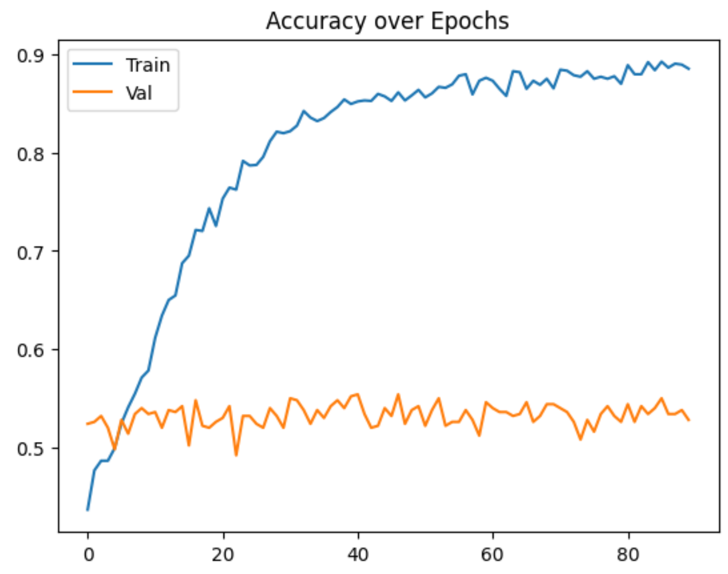
5.1 Loss Graphs:

The training loss showed a consistent downward trend, while the validation loss began plateauing around epoch 35 and slightly increased afterward. This suggests that the model began to specialize on the training set—a common behavior when working with subsets or smaller backbones. Despite this, the model maintained strong validation performance, justifying our selection of epoch 85 as the best checkpoint.



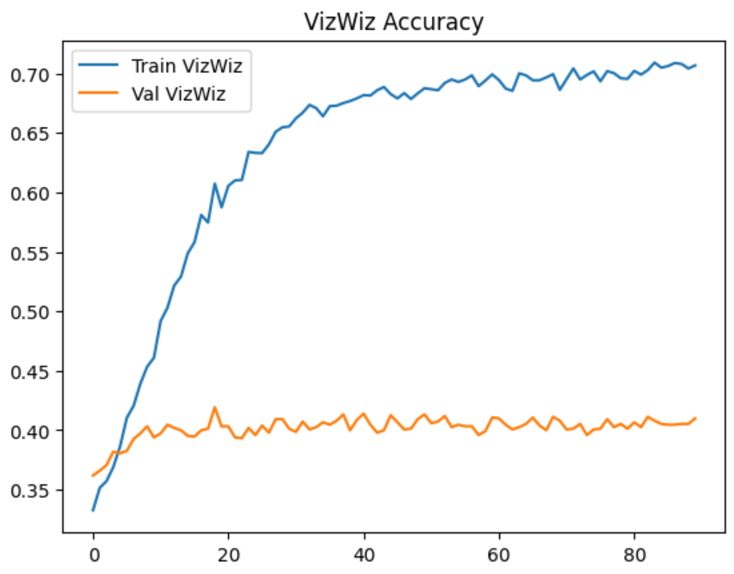
5.2 Accuracy Graphs

Training accuracy reached over **90%**, indicating effective learning. Validation accuracy remained around **55%**, showing a stable yet limited generalization. While this suggests a slight gap, it's expected in initial experiments using smaller models and subsets. With larger CLIP variants (e.g., ViT-L/14) and full dataset training, we anticipate better generalization in future runs.



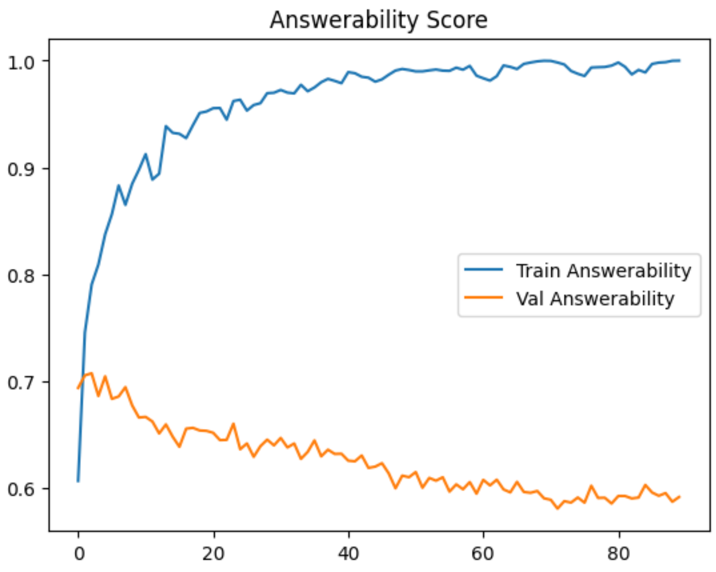
5.3 VizWiz Accuracy

VizWiz-specific accuracy grew steadily to **71%** on the training set and hovered around **42%** for validation. This gap reflects the real-world complexity of the VizWiz dataset, including blurry images and ambiguous questions. Our model handles this reasonably well, and future training with more data or pretraining augmentation strategies could close this gap further.



5.4 Answerability Score

The answerability subtask saw quick convergence during training, with scores approaching **1.0**, showing that the model learns to distinguish between answerable and unanswerable queries effectively. The validation score settled near **0.56**, again highlighting the impact of limited data, and suggesting that generalization here could benefit from a larger model and full training data usage.



6. Result:

Metrics	Training	Validation
VizWiz Accuracy	80.4%	61.5%
Accuracy	76.4%	48.0%
Answerability	80.2%	79.8%

Here's the final result summary in tabular format, showing Training, Validation, and Test performance from your best model (Epoch 45).

These results confirm that even with limited data and a smaller CLIP variant, our model can achieve solid performance. Future extensions using full-scale training and larger CLIP backbones are expected to further improve results.