**Visual Question Answering with Multi-Model Architectures**

**Data Pre-Processing:**

**Data sources:**

We have used the VizWiz QA dataset for the experiments. The dataset will include the images taken by the visually impaired individuals along with the natural language questions and the human-annotated answers.

**Image Preprocessing:**

All the images have been fetched from the URLs from the VizWiz dataset. Preprocessing step includes:

**Resizing the image to 128x128**: Ensure the standardized input size for the model and reduces the computational cost.

**Normalization with Mean and Std:** This helps to stabilize the training.

Mean : [0.485, 0.456, 0.406] std: [0.229,0.224,0.225]

**Converted to PyTorch tensors**: This is required for the model compatibility.

**Question Preprocessing:**

**Lowercase**: Ensures consistency which treats for example “What” and “what” as the same token.

**Expanding contractions:** Converts for example “what’s” to “what is” for better tokenization.

**Removing special characters:** This eliminates the noise from the data.

**Tokenisizzing using Spacy:** Breaks the text into meaningful units.

Removing the stopwords, punctuations and applying the lemmatization: Reduces the variability in the dataset and retains the core semantics.

We have used the GloVe pre-trained embeddings to convert the tokens into vectors and the mean vector was used to represent each question. This design choice will ensure that even varying length questions have the fixed length representation. This was chosen due to its balance of semantic richness and computational efficiency also avoids the complexity in question pipeline line in the model which keeps the model lightweights and interpretable.

**Answer Preprocessing:**

**Selected the most frequent answer from the 10 responses**: This will ensure the consistency across the different annotators.

**Took top the 300 most common answers:** Which will cover frequent answers while keeping the classification task feasible.

**All others were grouped under an “other\_categories’ class:** This prevents the model from overfitting to the rare answers.

**Balanced dataset for answerability classification:**

This will ensure a balanced dataset for the answer classifier model with 1:1 ratio answerable to unanswerable questions. This was achieved using the custom sampling logic based on the “answerable” label. This choice will prevent the model from boasing towards frequently occurring unanswerable questions and helps to improve the model generalization.

**Dataset Splits:**

**Train:** 2000 samples

**Validation:** 300 samples

**Test:** 100 samples

These sample sizes have been chosen to balance the performance and computational efficiency. In particular 2000 training samples have been chosen to offer enough diversity for the model to learn the meaningful sentences while keeping the memorable usage manageable on available hardware. This choice has been influenced by the computational limits of training multimodal models and having the desire to maintain rapid iteration during the hyperparameter tuning. It also helps to mitigate the overfitting and also ensures the fast experimentation cycles.

**Multi Modal Architectures**

We have designed the two distinct models:

**Answer Classification Model:**

This is the binary classifier which predicts whether the question is answerable or not.

**Architecture Overview representing input and output dimensions**

**Image Pipeline**

Conv2d (3×128×128 →64×128×128), ReLU, BatchNorm, MaxPool (→ 64×64×64)

Conv2d (64×64×64 → 128×64×64), ReLU, BatchNorm, MaxPool (→ 128×32×32)

Conv2d (128×32×32 → 256×32×32), ReLU, BatchNorm, MaxPool (→ 256×16×16)

Conv2d (256×16×16 → 512×16×16), ReLU, BatchNorm, MaxPool (→ 512×8×8)

Flattened and passed through Linear (32768 → hidden\_dim)

Dropout has been applied to the last Fully Connected Layer.

**Question Pipeline**

Linear (50 → hidden\_dim), ReLU + Dropout

**Cross Attention:**

Multi Headed attention where the image is query and question is key/value.

**Fusion and Classification:**

FC(hidden\_dim → hidden\_dim), ReLU + Dropout

FC(hidden\_dim → 1)

**Input:** Image tensor (3x128x128), Question embedding (50-d GloVe vector)

**Output:** Single scalar logit which is interpreted via sigmoid for binary prediction.

**Answer Generation Model:**

This model will predict the one of 301 classes which are 300 common answers and other\_category answers.

**Architecture Overview representing input and output dimensions**

**Image Pipeline:**

**ResNet18 (pretrained):** First several layers are frozen for efficiency and to retain the general image features. Only the last 4 layers were fine-tuned.

**Original ResNet FC replaced:** fc.in\_features -> hidden\_dim

Fully Connected layer(hidden\_dim → hidden\_dim×2) → ReLU + Dropout

Fully Connected layer(hidden\_dim → hidden\_dim×2) → ReLU + Dropout

**Question Pipeline:**

Fully Connected (50 → hidden\_dim×2) → ReLU + Dropout

**Cross Attention:**

Multi Headed attention where the image is query and question is key/value.

**Fusion and Classification:**

Concatenate the image features and attended features (2×hidden\_dim)

Fully Connected (2×hidden\_dim → hidden\_dim), LayerNorm, ReLU, Dropout

Fully Connected (hidden\_dim → hidden\_dim), LayerNorm, ReLU, Dropout

FC(hidden\_dim → 301)

**Input:** Image tensor (3×128×128), Question embedding (50-d GloVe vector)

**Output:** 301 class scores which are soft,axed during evaluation.

The models handles the different outputs by adapting the final layer and the loss function to the task:

For answerability classification, as a single output neuron has been used with a binary cross entropy loss.

For answer generation, the final layer has the 301 output neurons, one for each class and it uses cross-entropy loss suitable for multi-class classification.

Dropout layers have been added across both the models to reduce overfitting and improve the generalization. A dropout of probability of 0.1 was chosen as it priovies the good trade-off between regularization and training stability, where higher values like 0.3 or 0.5 which will degrade performance by underutilizing the learning representations.

ReLU activation has been consistently applied after all the non-linear transformations to introduce non-linearity while avoiding the vanishing gradients. This has helped to maintain the strong gradient flow during the training and accelerated convergence.

**Model tuning and Optimization**

During each trial of the Optuna hyperparameter tuning, the model weights with the best validation accuracy for that trial has been saved. At the end of all the trails, the best overall model across the trails is selected and its weights are saved. The final model was then used for the testing.

**Answer Classification Tuning**

**Optimizer:** Tried with SGD (momentum of 0.9 ) and Adam, AdamW (weight decay of 1e-4)

**Learning rate:** Sampled from log uniform range 1e-3 to 5e-3

**Hidden dimensions:** [128,256,512]

**Batch Sizes:** [32,64]

**Best Configuration:**

**Optimizer:** Adam

**Learning rate:** 0.0048

**Hidden dimensions**: 512

**Batch Sizes:** 32

**Trend Observed:**

Larger dimensions have consistently yielded better results by providing the richer feature representations.

The Adam optimizer has shown the faster conference and more stable performance when compared to SGD and AdamW.

Validation accuracy was sensitive to batch size where smaller batch sizes improved the generalization.

Models trained on unbalanced data tended to predict the either answerable or unanswerable more often. Introducing the class balance has improved the robustness and calibration.

Best Accuracy on Validation: 60%

**Answer Generation Tuning**

**Optimizer:** Tried with SGD (momentum of 0.9) and Adam, AdamW (weight decay of 1e-4)

**Learning rate:** Sampled from log uniform range 1e-3 to 5e-3

**Hidden dimensions:** [256,512]

**Batch Sizes:** [16,32]

**Number of Heads:** [2,4]

**Best Configuration:**

**Optimizer:** Adam

**Learning rate:** 0.00249

**Hidden dimensions:** 512

**Batch Sizes:** 16

**Number of Heads**: 4

**Trends Observed:**

ResNet18 has helped in improving the performance due to its strong feature extraction ability and also transfer learning advantages.

Smaller batch sizes have helped the model to learn between which is likely due to more frequent updates.

Increasing the number of attention heads from 2 to 4 has improved the results which is possibly due to the better representation of multi-modal relations.

Moderate learning rate has helped the model to perform well.

Best Accuracy: 55%

Momentum of 0.9 has been used with SGD to accelerate convergence and reduce the oscillations.

Weight decay of 1e-4 was chosen to regularize the model and reduce the overfitting by penalizing the large weights.