Neural Networks LAB 3

1) Details of the data pre-processing step. How was the dataset pre-processed? Why did you make the design choices you did? How much training data was used and why?

   Data to be used consisted of both Images and questions which need to be merged and processed according to the results expected. I preprocessed images and text data separately using the following methods:

   a. Image preprocessing:
      i. I used torchvision.transforms to apply the transformations to the images.
      ii. I resized the images to 128*128, this ensures all images have the same reduced size so that it can help in improving the computation speed.
      iii. I used a data augmentation technique called Random Horizontal flip with a probability of 0.5, This introduces variations in the data making the model robust when working with different orientations.
      iv. I normalized the data set with the stand mean and std deviation values of the ImageNet Dataset. This helps in stabilizing the model and thus in better model convergence.
      v. I removed variations in the brightness, contrast and saturation to improve generalization using the Color Jitter function.
      vi. Converted the images into tensors.

      Reasons for the design choices:

      Resizing helps in ensuring all images have uniform input dimensions, augmentation and Color variations helps in preventing overfitting and improves generalization. Normalization helps the model efficiently learn the data which now is comparable on a similar scale. The tensor conversion is essential for using it in a model.

   b. Text preprocessing:
      i. Tokenization to split sentences to words.
      ii. Converted all data into lowercase for consistency.
      iii. Removed stop words like “the”, “is” etc., since they do not contribute to the meaning of a sentence. Used the nltk’s stop word list.
      iv. Used lemmatization to convert a word to its most simple form, this reduces dimensionality of the questions and improving consistency.
      v. I converted the questions into numerical vectors of size 50 using the GloVe 50D word embeddings. I find embeddings of each word in a

question using the downloaded embeddings, if no embedding is found I take a 0 tensor. I find the mean of all the embeddings to get the final representation of the question as a 50-length vector.

Reasons for the above preprocessing steps:

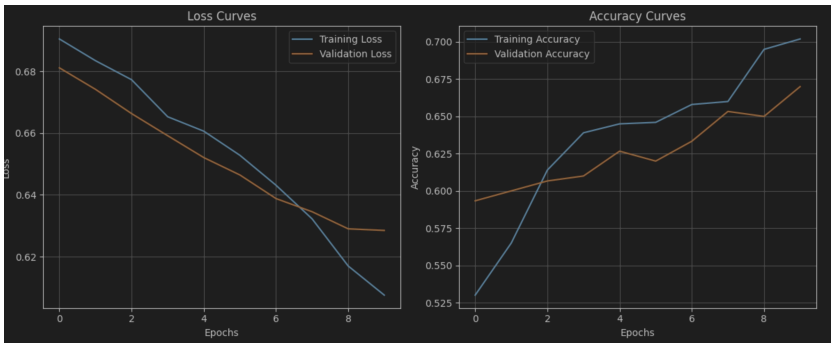
GloVe embeddings 50D was used for the size of the embedding which helps in efficient computation, and the dataset is preferred because of the semantic richness of it. Mean pooling helps in reducing the questions of variable length to fixed size representations. Stop word removal and lemmatization helps in reducing noise and helps model identify patterns better.

Training Data Used Comparison and Reasons:

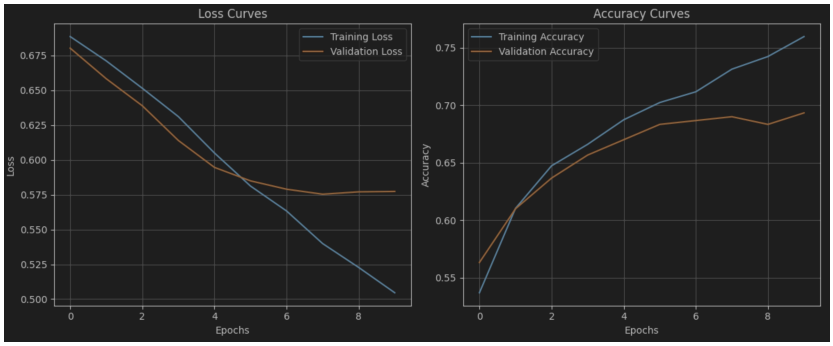
I used 18000 images and questions as the dataset for the first challenge, I also used equal amount of data for 0, 1 class in the challenge 1, to remove class imbalance.

For challenge 2, I used only 8000 images, this is mainly due to computation reasons, Since the model required a huge amount of time to train and test.

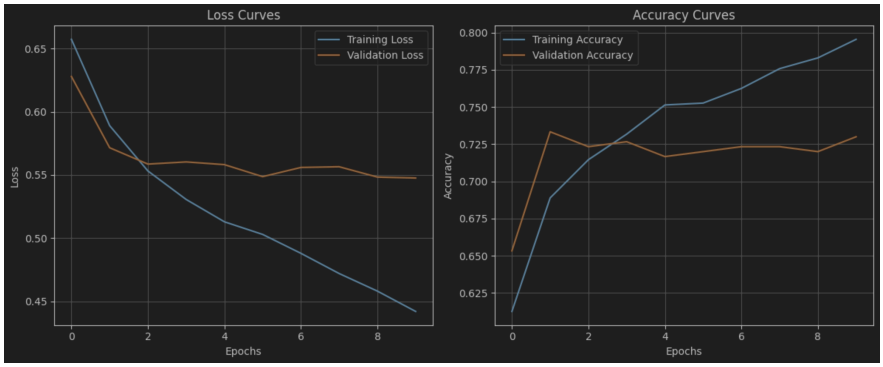
With 1000 samples



With 3000 samples



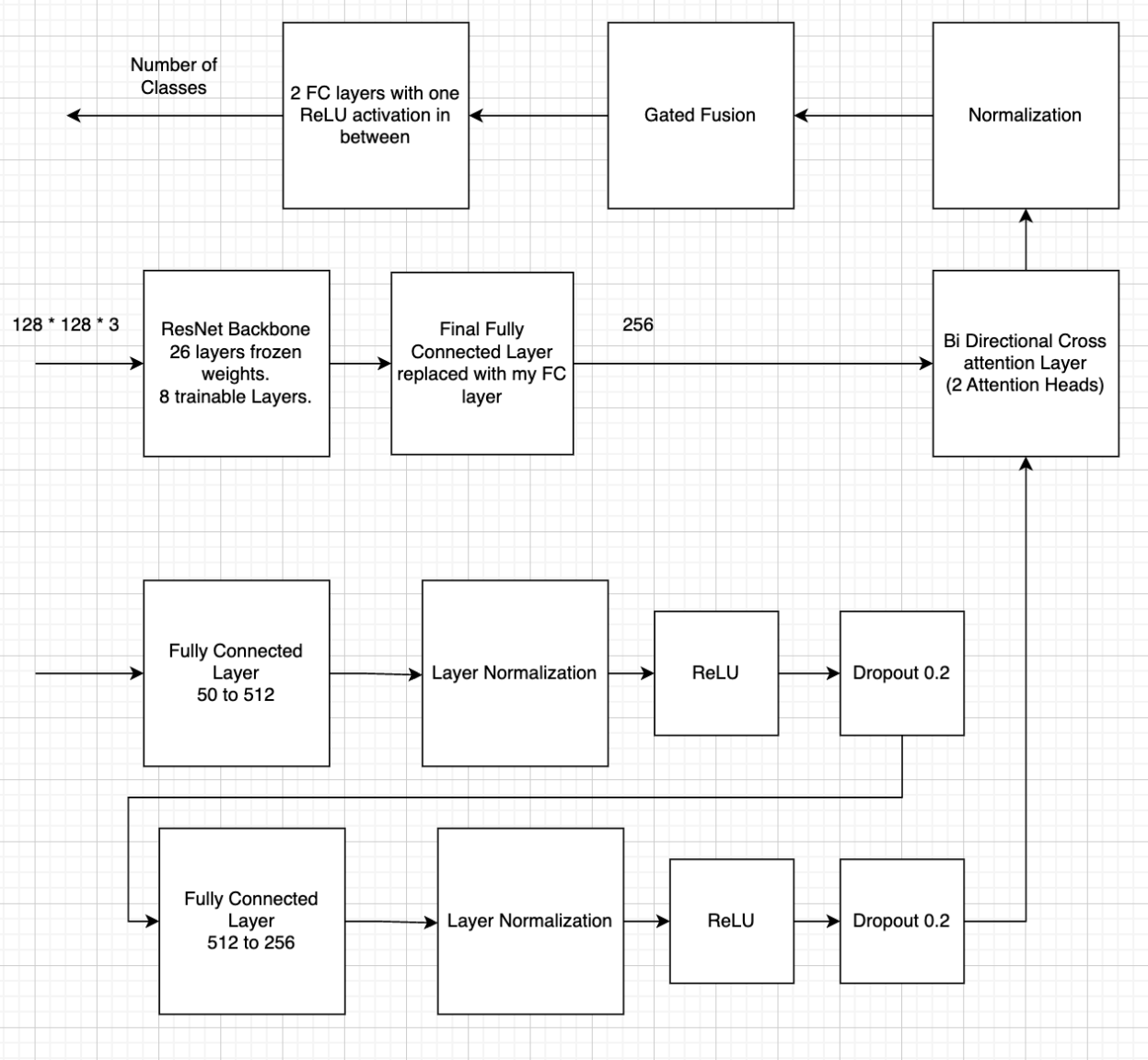
With 18000 samples



As we can observe highest validation accuracy with 1000 samples is 67.5%, with 3000 samples it is 69% and that of 18000 samples is 72.5%. Also, this means that the model is generalizing better with more data. And using the entire dataset will result in better accuracy. For reducing the training time, I have made some compromises on the amount of data used for training.

2. Details of the multi-modal architectures. Which types of layers did you use? How do the models incorporate both modalities? How do they handle the different types of output?

The Multimodal architecture consists of the following blocks:



First, we process the Images using a ResNet34 Backbone to extract features of the Image. The last first 26 layers are frozen, and the rest 8 layers be trained on the VizWiz Dataset. Final FC layer of the RestNet is replaced with a FC layer which converts it to a hidden dimension width of 256. The text layer consists of 2 blocks with each having a FC Layer, Layer Normalization, a ReLU activation and a dropout layer. The two features of width 256 is used to calculate cross attention. Between images and text and text and images to find complex relationship between the two modalities. With first layer having the Q values as Image features, K, V as text features, the second layer with the opposite setup.

Then the attended features are normalized and are fused using Gated Fusion after concatenating the feature results. This is then passed through final 2 fully connected layers

to provide the output.

The reason for using the layers are as follows:

1) I used the ResNet because it uses CNN to extract useful features from Images. Have a few layers unfrozen so they can learn features from VizWiz Images.
2) I used Feed forward layer to process the question embeddings and added layer norm, ReLu activations and Dropout for better stability.
3) I used Bidirectional Cross Modality Multiheaded Attention layers to get the image-to-text relationship and text-to-image relationships. This helps in focusing on the parts of image which is important given the text and vice versa.
4) I used the sigmoid gated fusion to blend the images and text features after concatenating them, so that no information is lost. Gated Fusion dynamically weighs the contribution of the text and the image features.
5) Final 2 layers for normal classification to produce number of classes as result.

Model Fusing the Modalities:

The model incorporates both modalities by first converting the image and text features to the same hidden dimension so they can interact properly. Then, it applies two multi-head cross attention layers — one where the image attends to the text, and another where the text attends to the image. This bidirectional attention helps each modality focus on what's most relevant in the other. After that, the outputs from both attention layers are normalized for training stability.

To fuse the attended features, they're concatenated and passed through a gating mechanism. This gate learns how much importance to give to each modality — whether to rely more on image features or text features — and produces a balanced, combined representation that's used for the final classification.

Handling the outputs:

For the challenge one, the model is designed as a binary classifier, the output is a single logit, which is passed through a Binary Cross Entropy with Logits Loss function to predict weather a given answer is answerable or no.

For challenge two, the model is made into a multi class classification problem by finding the most frequent top n answers and predicting the probability of these answers, with the rest of the answers falling into the other_category class. So, this uses Cross Entropy loss to predict the most probable answer in the set of top n classes.

3. Details of model adjustment and hyper-parameter tuning using the validation set. Did the model development process reveal interesting trends about architecture design? Which hyper-parameters were the most important? How did you decide on the final versions?

Model adjustment and Hyper parameter tuning using validation set:

The model development required both architectural changes and hyper parameter tuning to ensure that the model was not overfitting, data efficient and yields the best accuracies on both sets.

Hyper-parameter tuning:

Initially I fixed values for learning rate (LR), weight decay and batch size based on the experimental results.

- Learning Rate (LR): 0.0005 led to a stable training. This allowed the model to converge well without overfitting and with validation accuracy increasing constantly.
- Batch Size: Changing batch size had negligible effect on performance. I selected 32 since it was a nominal value.
- Weight Decay: A small weight decay of 1e-5 was sufficient to improve generalization.

I also experimented with number of attention heads in multi head attention layers. Increasing it to 4 or 8 does not have significant effect on the output, so I fixed it to 2 heads to reduce the parameter count and improve the training speed.

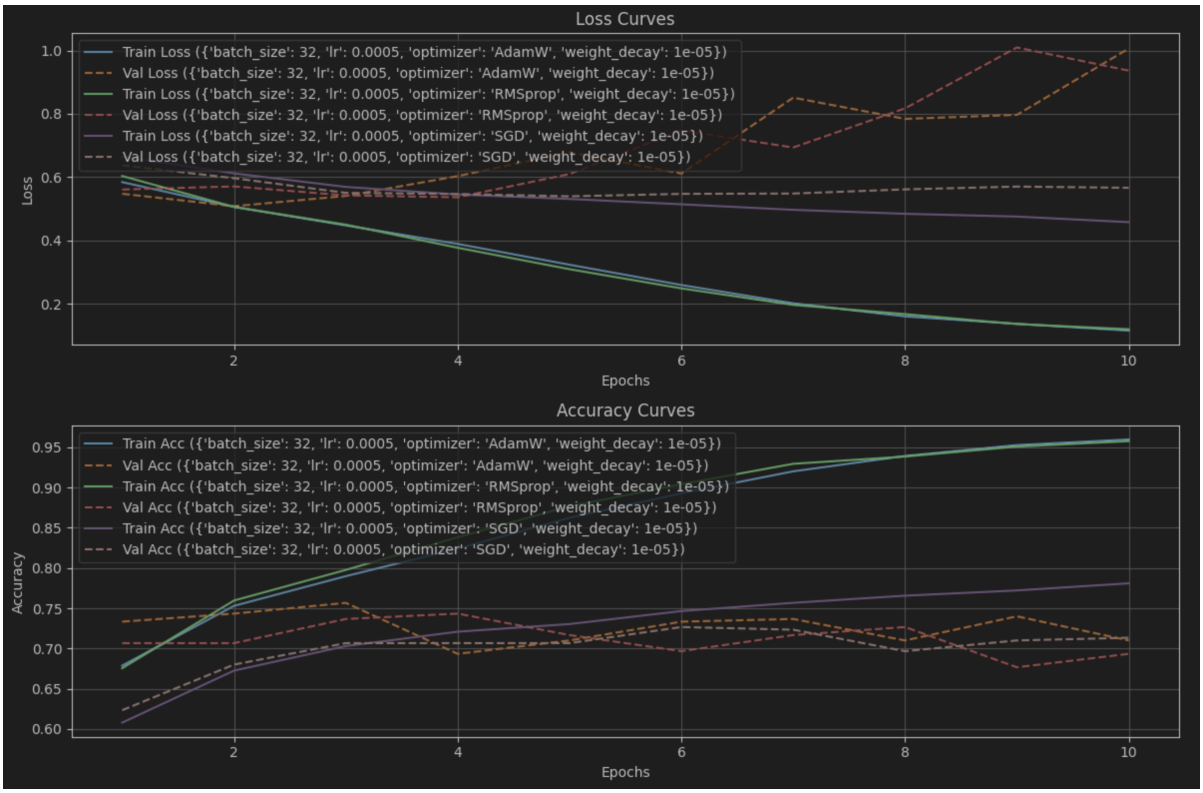
Architectural Trends:

The Architectural changes had significant impact on the accuracy.

- ResNet Backbone: Without any visual backbone the accuracy was around 52% close to random guessing. With the ResNet34 the validation accuracy increase was 10%. By freezing the first 26 layers and allowing fine tuning in the last 8 layers, the model captured image features better.
- Feature Fusion: Initially, the simple concatenation of image and text yielded poor results. Replacing it by bidirectional multi head cross attention layers (image-to-text and text-to-image) led to a 5% increase in the validation accuracy, because of the better interaction between the modalities.
- Gated Fusion: The gated fusion mechanism helped combining the attended image and text features. This added a 3% gain in the accuracy by balancing the contribution of both modalities.

Most influential Hyper parameter selection:

The optimizer selection has the most influence in the training of the model. I compared SGDM, AdamW and RMSprop.



As seen in the Image the use of AdamW and the ResNet makes the model to overfit. Both the models achieve a training accuracy of 97% and training loss very close to 0, but the validation accuracy is very low, and the loss is very high resulting which is a clear sign of overfitting. Although the model with SGDM has a low amount of convergence, it generalizes better which can be seen with the validation accuracy and loss, and so I retained it for this.

Fixing the Model:

The only problem I had with the model was it was overfitting. All other hyper parameter tuning and architectural changes helped in improving the accuracy, but the stability of the curve and overfitting was a problem. After changing the optimizer, the model worked as intended. After this, increasing the model complexity or adding layers resulted in the model performance depleting with a reduction in the accuracy by 10%. So, I fixed this model for evaluation and to participate in the challenge.

I finalized the model using **SGDM**, a learning rate of **0.0005**, **weight decay of 1e-5**, **batch size of 32**, and **2 attention heads**. The architecture included **ResNet34 with partial fine-tuning**, **bidirectional attention**, and **gated fusion**, which consistently produced the best

validation results. I also saved to best model parameters that yielded the highest validation accuracy during training.

Using these the model achieved:

Challenge 1: Validation Accuracy 73.33%. Training Accuracy 68.89%

Challenge 2: Validation accuracy of 36%. Training Accuracy 40%.