Milestone 3 Project Report

Current State of the project:

Integration of Text Model:

We optimized our model for mobile using mobile_optimizer provided by PyTorch Mobile. We then converted our optimized text model to TorchScript to make it compatible with Android. To convert the String entered by the user into Tensors understood by the model, we converted the model's vocabulary as a CSV file and placed the model.pt file and our vocabulary.csv file into the 'assets' folder.

We accessed the model from the 'assets' folder by using the AssetManager. Got the input entered by the user, tokenized it into different words, and translated the words into an array of indices that could be understood by our model, making sure that unknown words were classified into the '<unk>' tag. We then converted our array of indices into a Tensor, passed it to the model, which gave us a tensor of Float values corresponding to our classification weights. Eventually we used these weights to classify the caption entered by the user into happy, sad, scared or angry, and displaying the corresponding image with the caption over it.

Improvements in the Text Model:

We realised that our classifier had a numerical bias towards 'sad' and misclassified a number of things we inputted. To work around that, we divided the 'sad' category into 'sad' and 'scared'.This reduced the numerical bias of 'sad' and allowed better prediction. We also devised writing our vocabulary to a file to be used in our app and saving the optimized model.pt file for our ML model.

Integration of Image Model:

We generated a model.pt file from our image classification model to be compatible with Android. We did this by quantizing our model on the Google Colab notebook to be able to integrate it with our Java code. Then, we tested our model on a sample image part of the 'assets' directory in 'Cats/app/src/main'.

Improvements in the Image Model:

After testing with a sample image, we proceeded to use random Google images to further test our model. While our model did classify around 4 out of 10 of the images we used in our random test dataset at first, the accuracy can be improved by fine-tuning the app for Milestone 4. Currently, we are debugging and optimizing our model - we may decide to reduce the number of categories to 2 classes (happy and sad) in order to improve the outcomes of our meme generator.

App Fine-tuning:

We fixed the “make meme” button in the Image Upload activity. We also brainstormed potential names for the app. Furthermore, we are looking into implementing bonus features like the integration of Google Lens.

Current Challenges and Bottlenecks:

Text Classification Model:

The text classification model performs reasonably well. We decided to go with ngrams = 1, partly to keep the vocabulary length short and partly to increase the weight of an individual word. But in doing so, we did compromise on the ‘context’ aspect of a sentence. With our current model, if a sad sentence with a number of happy words is inputted, it gets classified as happy.

Image Classification Model:

The image classification model has a ~80% accuracy on google colab notebook as of now. We also tried to reduce the number of categories to 2 (happy and sad respectively), which gives us ~87% accuracy. But the model seems to classify images into “angry” on our app when this category is included. We plan on comparing the input tensor size or/and scores of each category on notebook and the app to debug the integration code. Currently, our integrated model correctly classifies between happy and sad cats, but we are working on incorporating the ‘angry’ cat class as well as the ‘sleepy’ category eventually.

Work Distribution:

- Christy: Worked on integrating the Image Classification model with the App in collaboration with Fariha. Made the Image classification model compatible with Android.

- Fariha: Worked on integrating the Image Classification model with the App in collaboration with Christy. Tested the image classification model with different parameters and quantized it to make it compatible with Android.

- Jayant: Worked on integrating the Text Classification model with the App in collaboration with Savitoj. Improved the Text Classification model, created a vocabulary file to access in the app, converted the model to TorchText to make it compatible with Android.

- Savitoj: Worked on integrating the Text Classification model with the App in collaboration with Jayant. Improved the Text Classification model, created a vocabulary file to access in the app, converted the model to TorchText to make it compatible with Android. Fixed the “Make Meme” button for the image upload activity.