MeMeow

Team Rubeosaurus
Jayant Bhardwaj, Savitoj Sachar, Fariha Shaikh, Christy Zhang

Submitted to
Mieszko Lis
CPEN 291 Computer Engineering Design Studio I

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. PROJECT DESCRIPTION

Team Rubeosaurus spent a month bringing to you MeMeow, an ML-based Cat Meme Generator app for Android. The application allows you to convert the image of a cat into a meme. Our Image Classifying model recognizes the sentiments of the cat in order to bring you an interesting meme. But not just that, we also allow you to make a meme from text (works well for people without cats). Our Text Sentiment Analysis model classifies your text into one of four sentiments - happy, sad, angry, and scared - and creates a cat meme based on that. Either way, you get cat memes and are guaranteed peak entertainment!

2. WALKTHROUGH EXAMPLE

As the user embarks on their journey with MeMeow, they are greeted by our aesthetically pleasing main page (see Figure 1), offering them the option to make memes from text or images. (Clicking on the button on the bottom right corner will redirect the user to an information page, see Figure 2.)

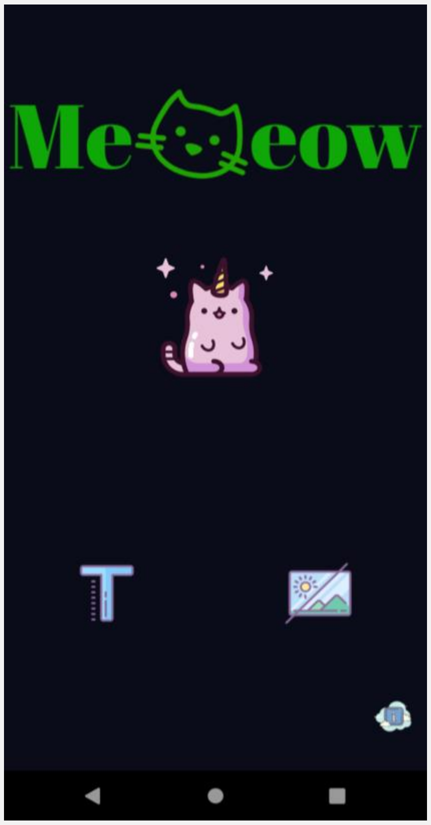
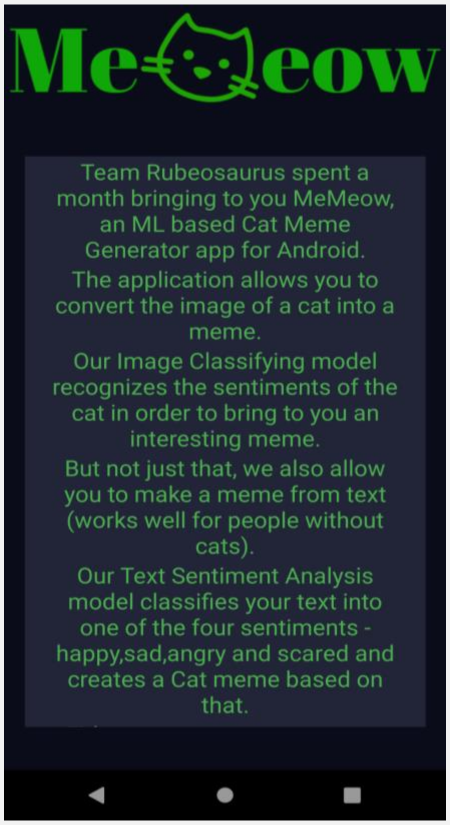


Figure 1. Main page



Team Rubeosaurus spent a month bringing to you MeMeow, an ML based Cat Meme Generator app for Android.
The application allows you to convert the image of a cat into a meme.
Our Image Classifying model recognizes the sentiments of the cat in order to bring you an interesting meme.
But not just that, we also allow you to make a meme from text (works well for people without cats).
Our Text Sentiment Analysis model classifies your text into one of the four sentiments - happy,sad,angry and scared and creates a Cat meme based on that.

Figure 2. Info page

2.1 Making a Meme From an Image:

If the user wishes to create a meme from an image, they can click on the icon in the bottom right corner of the main page (see Figure 1); this will redirect them to the image upload page (see Figure 3). The user can then press the 'Select Image' button on the image upload page to choose an image from their gallery or Google Drive. The selected image will then be displayed on the image upload page to preview. A sample image input is shown below (see Figure 4).

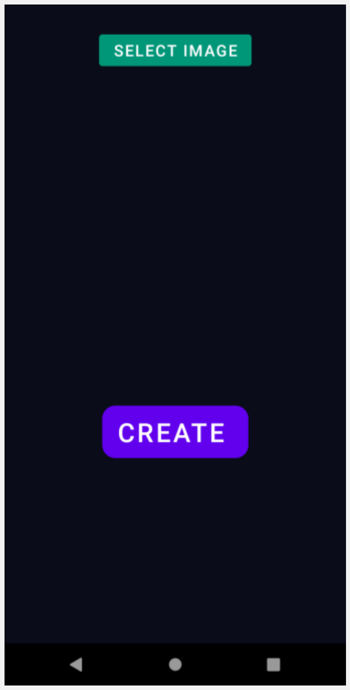


Figure 3. Image upload page

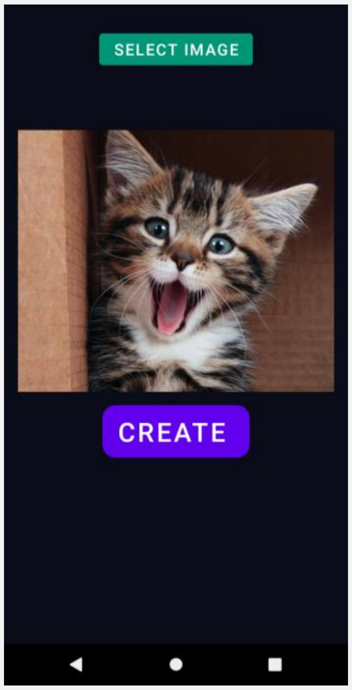


Figure 4. Preview of selected image

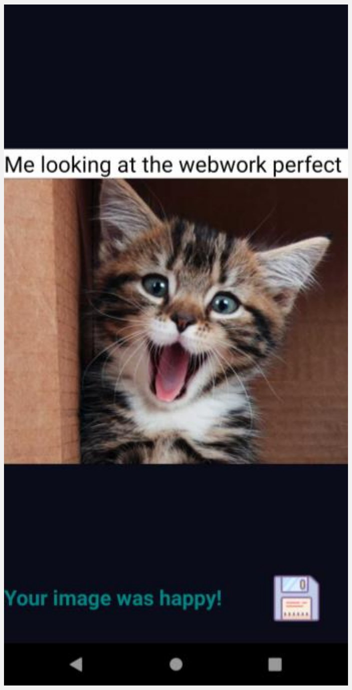


Figure 5. Meme output page

Once the image is ready, the user can press the 'Create' button on the image upload page (see Figure 4). This enables the image classification model on the backend to analyze the input image and classify it into one of two categories: 'happy' or 'sad. The output meme will then be displayed on the meme output page after the image input is paired with a matching caption (see Figure 5). Finally, the user can press the save icon on the bottom right corner of the meme output page (see Figure 5) to download the meme and save it to their gallery.

2.2 Making a Meme From Text:

If the user wishes to create a meme from text, they can click on the icon in the bottom left corner of the main page (see Figure 1); this will redirect them to the text entering page (see Figure 6). The user can then enter text (that will be matched with an image) and press the 'Create' button on the text entering page (see Figure 6). This triggers the sentiment analysis model on the backend to classify the text into one of four categories: 'happy', 'sad', 'angry', or 'scared'. As before, the output meme will then be displayed on the meme output page after the input text is paired with a matching image (see Figure 7). The user may then press the save icon on the bottom right corner of the meme output page (see Figure 7) to save the meme to their gallery.

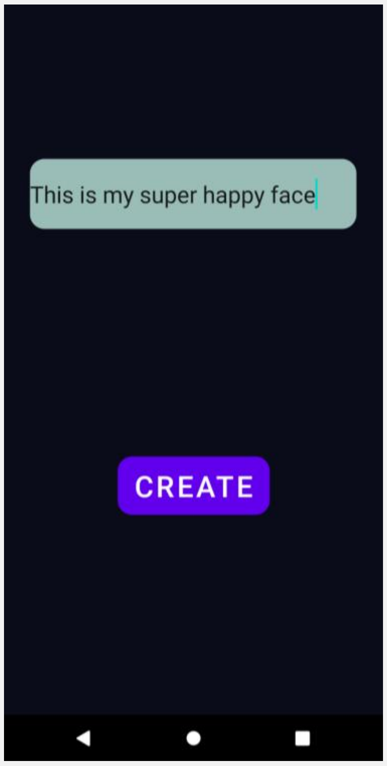


Figure 6. Text entering page

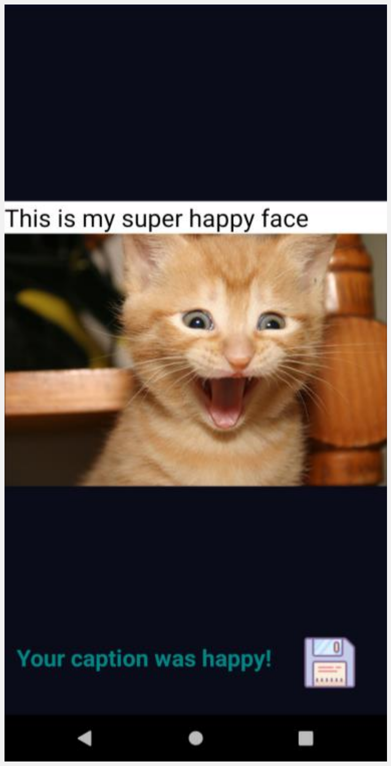


Figure 7. Meme output page

3. DATA FLOW BETWEEN SYSTEM COMPONENTS

Our project consists of three components: an image classifier machine learning model, a sentiment analysis machine learning model, and an app development module that integrates all other subparts of the system together. See Figure 8 for a visual representation of the data flow.

The inputs to both our image classification model and sentiment analysis are the datasets we used to train and test our model. For our image classifier, the dataset comes in the form of images of happy and sad cats. On the other hand, the dataset for our text classifier involves text categorized into six categories: fear, surprise, love, joy, sadness, and anger. The outputs of both our image classifier and text classifier are the models we optimized after experimenting with different parameters like learning rate scheduler, step size, etc. These models are then used as inputs to our app module , along with the image or text provided by the user. After pairing the image/text with text/image from our database, the final output of the app module is a cat meme.

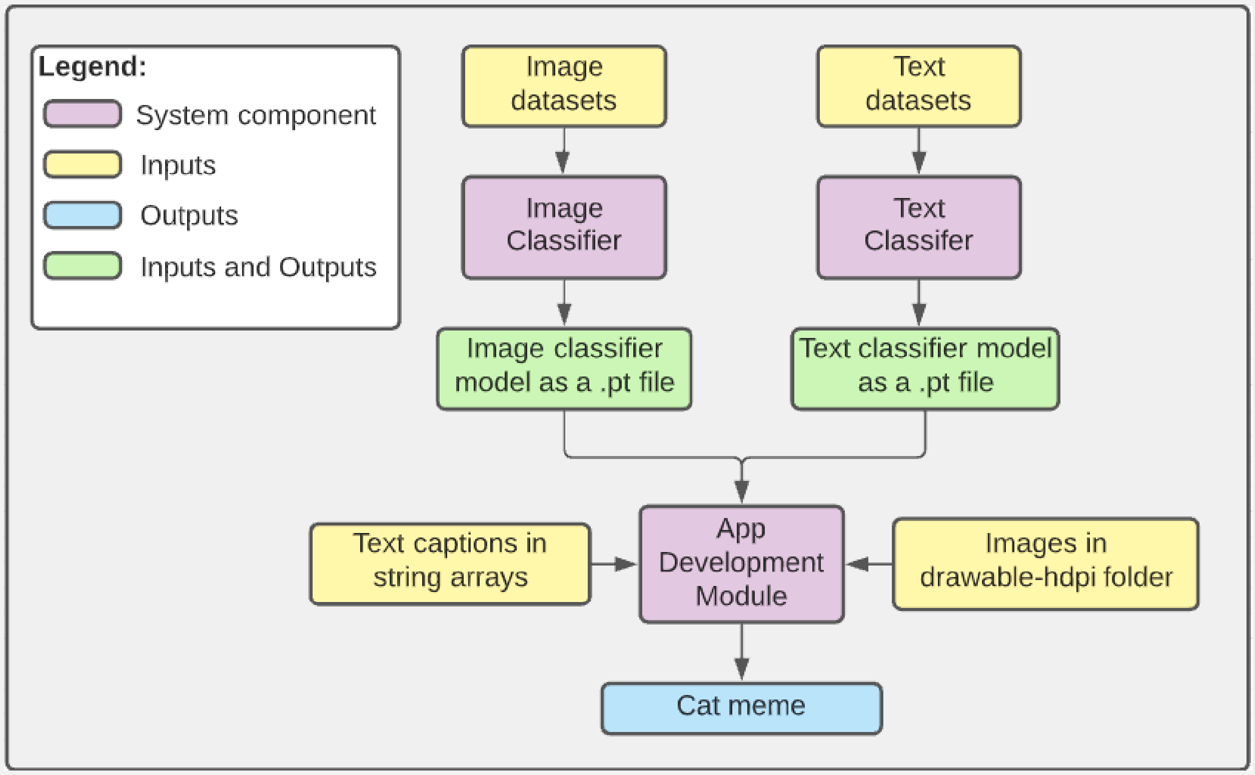


Figure 8. Diagram of data flow

4. DESCRIPTION OF SYSTEM COMPONENTS

Our implementation of MeMeow consists of three parts, as mentioned above: an image classifier, a text classifier and an Android app development module that was integrated via Android Studio. In this section, we will discuss how we implemented each of these components in detail.

4.1 Image Classifier

The dataset we used to train our image classification model involves two categories: ‘happy’ and ‘sad’. We collected 300 real cat images for each of our four initial categories: ‘happy’, ‘sad’, ‘angry’, and ‘sleepy’. [2] The main algorithm applied to our model was very similar to assignment 2. We used resnet18 as the pretrained model and added a fully-connected layer to classify cats as either happy or sad. We employed Cross Entropy Loss, the AdamW optimizer, and the stepLR scheduler throughout the training process.

We also created a ‘transferLearning’ class to help with testing different sets of parameters. After attempting to adjust the values of our optimizer/scheduler parameters and using data augmentation (through image transforms like RandomGrayscale and HorizontalFlip) to enrich our dataset, we managed to reach an accuracy worth 87.54% for our image classification model. By comparing the loss/accuracy results for our 4-category dataset with those of our 2-category dataset, we decided to deviate from our original plan of having four classes since the accuracy was lower than what we wanted. The parameters for our final optimized model are as follows: a learning rate of 0.0001, a momentum factor of 0.8, a learning rate scheduler of 0.001, a batch size of 32, and a step size of 0.01.

4.2 Text Classifier

For our sentiment analysis model, we used a text dataset involving six groups of emotions: fear, surprise, love, joy, sadness and anger) sourced from kaggle. [3] We also used the Pandas framework to handle dataset manipulation as it was more convenient for our objective. First, we removed all entries associated with the surprise and love category and mapped anger, fear, joy, and sadness to angry, scared, happy, and sad respectively. Furthermore, since the original dataset came in separate files for testing, training, and validation, we changed the text file to a single CSV file. As a result, we obtained a full dataset with four classes of emotion: angry, happy, sad, sleepy, and scared..

To prepare the dataset for the training process, we created a dataset class that would handle the CSV data by tokenizing the captions and creating a dictionary.

We then used text embedding to represent the data as vectors and applied the LSTM model [4] onto our data in order to account for the overall context of the input. We also created functions to test the accuracy of our model and predict the classes of some specific captions. This confirmed that our model is at least 90% accurate and demonstrated that it works for the course-related meme captions we have too.

4.3 App Development Module

We began by creating 2 activities alongside the main activity for our app development module; one for image uploads and the other for text inputs. As we made some progress with our ML models, we added more activities and features to the app. First, we implemented the functionality for making a meme from a given image. We created a set of 20 captions (10 captions per emotion) from which one would be randomly auto-generated based on the classification of the image input by the user. Similarly, we collected 80 images (20 images per class) to be paired with the caption input by the user for our sentiment analysis model.

We then optimized our model for mobile using the mobile_optimizer module provided by PyTorch Mobile. We transformed our optimized models using TorchScript to make it compatible with Android and placed the model.pt files (as saved from our notebook) into an ‘assets’ folder within our app directory. Finally, we created an array of strings for our predefined captions and images and included the latter in the drawable-hdpi folder of the app.

After integrating the ML models with our app by converting the images to bitmaps and applying tensor manipulations to them, we focused on the overall app design and user interface. We also worked on resolving issues we had with resetting the bitmaps from our image upload activity to accommodate another input image if the user wishes to use the ‘back’ button to upload a new image. Next, we changed the color scheme of our app, added new buttons and textViews that allow the user to download the output memes and see the category their image/text was classified into. Finally, we improved the overall design of our logo and icons.

5. CHALLENGES

During the completion of this project, we faced a few challenges while trying to implement our cat meme generator. In this section, we discuss some of the issues we had with each component, and how we resolved them to improve overall accuracy and ensure proper integration.

5.1 Image Classification Model

The main challenge we encountered with our image classifier was the integration of our model. Although we solved our initial low accuracy and high test loss by using data augmentation and applying transforms to our images, after importing the model to our app, there was still a strong bias towards the ‘sad’ class. Upon closer inspection, we noticed that this was because the standard deviation values being applied to our normalized model on the app were different to the ones applied to our transform on Google Colab. Thus, we resolved this by ensuring consistency between both values and reducing the number of classes to two categories to preserve accuracy.

5.2 Sentiment Analysis Model

An issue we had with our text classifier was the model’s tendency to classify the input text into the ‘happy’ category. Initially, we were using ngrams = 1 to classify our text, but based on the feedback we received for our third milestone report, we decided to use an LSTM model to better account for the context in which the text was entered. This resulted in better accuracy and prevented our model from classifying text into the ‘happy’ category by default, greatly reducing train and test loss.

5.3 App Development

One of the biggest problems we had with app integration involved functionality for some of our buttons. When trying to re-upload a cat image to create a new meme, although the image preview reflected the image selected, the bitmap used to create the meme retained information from the previous image, meaning the output meme did not reflect the most recently uploaded image. To combat this, we updated our code so the bitmap and converted tensors would be reset every time the user clicked on the ‘back’ button to select a second image.

Additionally, a challenging design decision we made was to integrate our models into the app (frontend) rather than doing it via the backend. This allows the user to utilize our app without being connected to the internet, which we believe is important for making cat memes on the go. Although we do recognize that a disadvantage to this is that our app (.apk file) takes up more space, the pros of integrating it this way outweigh the cons.

6. OVERALL CONTRIBUTIONS

Jayant Bhardwaj:
Collected images for the app's image dataset. Trained and tested the text classification model. Integrated the text classification model with the App. Worked on changing the text classification model to an LSTM model. Fixed image model integration. Worked on training the image classification model. Designed and improved the app logo. Fixed meme layout to better fit all devices. Fixed the 45-character caption limit. Worked on making the back button work, UI upgrades, and general testing and debugging of the app.

Savitoj Sachar:
Setup the basic structure of the app. Created and cleaned datasets. Worked and reworked on the ML models. 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 through the app, converted the model to TorchText to make it compatible with Android. Fixed the "Make Meme" button for the image upload activity. Worked on making the back button work, some UI stuff and general testing and debugging of the app.

Fariha Shaikh:
Created and cleaned two initial datasets while collecting images for the app's dataset. Trained and tested the image classification model by optimizing parameters with data augmentation. Worked on debugging model integration for the image classifier. Generated captions for the app's make-meme-from-image functionality and worked on fixing the 'Create Meme' button for the image upload activity. Further improved on general UI features like color scheme, button and textView placement positions, and icon design. Edited, formatted, and proofread final report.

Christy Zhang:
Collected images for the app's dataset. Generated captions for the app's make meme from image activity. Worked on training image classification and image model integration. Worked on the training text classification model. Worked on changing the text classification model to an LSTM model. Fixed the text cut-off problem on the info page. Worked on general testing and debugging of the app.

REFERENCES

[1] Icons by Icons8, Source: https://www.icons8.com. [Accessed: 15-May-2021].

[2] Adrian Rosebrock, Python script to download bulk images from Google Images
Source: www.pyimagesearch.com [Accessed: 15-May-2021].

[3] Praveen, "Emotions dataset for NLP," *Kaggle*, 16-Apr-2020. [Online]. Available: https://www.kaggle.com/praveengovi/emotions-dataset-for-nlp. [Accessed: 15-May-2021].

[4] "L15.7 An RNN Sentiment Classifier in PyTorch," *YouTube*, 13-Apr-2021. [Online]. Available: https://www.youtube.com/watch?v=KgrdifrlDxg. [Accessed: 15-May-2021].