R Noah Padilla

Jonatan M Contreras

Semester Project: Digit Classifier Application

App Description:

This app will be able to classify digits from the numbers 0 to 9 found in images. The user will take a picture or select an image from their gallery of a digit and the app will tell the user what digit was in the picture. To classify the digit, the app will use an onboard pre-trained convolutional neural network (CNN) to predict the digit in the photo.After the user has been notified of what the digit was, this process can be repeated as many times as the user desires until they exit the app.

Development Challenges:

The main developmental challenges we faced were re-designs of our app design. Initially, we designed our application in a traditional model-view-controller layout with important classes that would be communicating between the models and the views. However, we discovered that Flutter helps abstract a lot of this information into widgets, helping remove a lot of complexity from the build. For example, we assumed we'd need control classes so that we could communicate between the camera, galleries, and the views to display images, as well as to pull the images to create objects with those images. However, Flutter supports this functionality through widgets and function calls. Thus, as we iterated on our design, we found ourselves removing classes that we simply didn't need. It resulted in our design being a lot simpler.

Another interesting design decision that we made was to remove the custom progress indicator. We had initially built a custom progress indicator (Appendix item a) so that we can display it to our users while classifications are being completed. However, after some hands-on experience with TensorFlowLite, we noticed that this library makes it so that classifications are quick, easy, and predictable. Thus, including a progress indicator would only clutter our design since it is not necessary. Thus, we decided to remove it from the final build.

Neural networks can predict a class for an image, even if the confidence level is low. TensorFlowLite allows for us to implement a threshold for which we don't want the

predictions to be below. For example, if its prediction for an image is the class 0 with a confidence level of 10%, and our threshold is 20%, then the prediction won't be made to begin with. We implemented our classifications such that, if the prediction confidence was below the threshold, then there was no classification made. Accordingly, the "Save Classification" button only displays on the UI if the confidence is above this threshold. If it is not, the application does not allow the user to save. We believe this to be the better implementation since the idea is to store actual classifications.

We also discovered that, in order to use pre-built machine learning models, you must first convert them to a .tflite file extension, which was not trivial. Tensorflow can be used for this, but it is important to note. Another important thing to note regarding this is that there are many .tflite models available online from other creators so it is possible for an app developer to find and use a model and treat it as a black box. This may be a positive or negative, depending on who you ask, but at least it's possible for app developers.

We also used the Gesture Detector widget in order to demonstrate classification detail to our users. We demonstrate the previous classifications in a GridView first, and if they want more information (e.g., to display the image larger or to know which classification was predicted for that image), then they can tap the image to display this Image.

Technical Lessons Learned:

We learned how to use TensorFlowLite so that we can use not just pre-trained convolutional neural networks (since that's the machine learning model deployed in our application), but any kind of machine learning model within our applications. Another important thing to note here is the viability of using machine learning with our applications. Machine learning has made a lot of interesting tasks doable now in general, and since it is viable to use these same models in Flutter via TensorFlowLite, we can now make applications that use these same models. This unlocks a lot of great possibilities for interesting applications.

While using TensorFlowLite was the main goal of this application, we also learned how to effectively use:

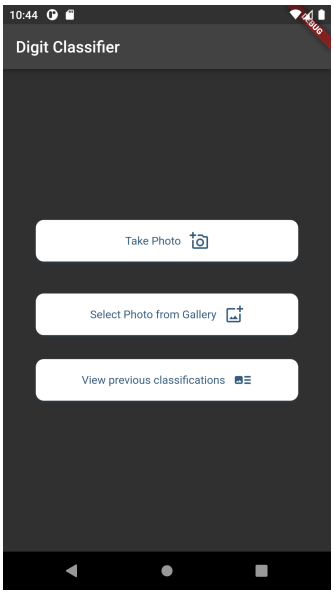
- SQFLite - We are using SQFLite for our local database of classifications.
- Camera & Gallery functionalities
- Containers and other widgets for aesthetically pleasing UI elements
- Custom Progress Indicators
- Gesture Detectors for tapping images in GridView and expanding details of classification.
- TensorFlowLite
- Conditional displaying of buttons and UI elements.

Project Complexity:

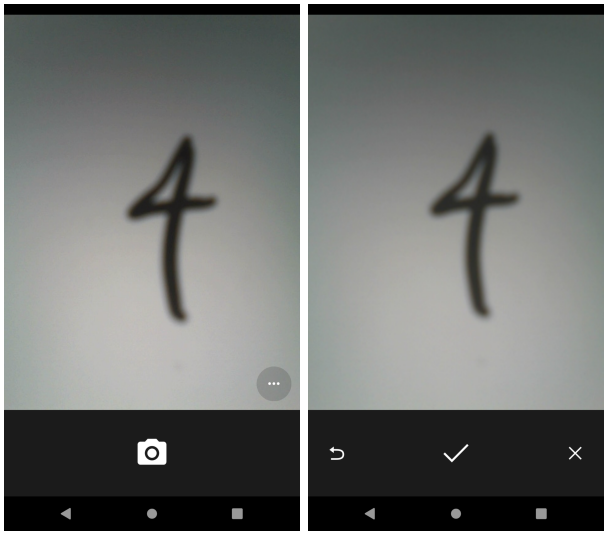
Number of Classes	**8**
Total Lines of Code	**628**
Flutter Features/Libraries	**See bullet list above**

Appendix

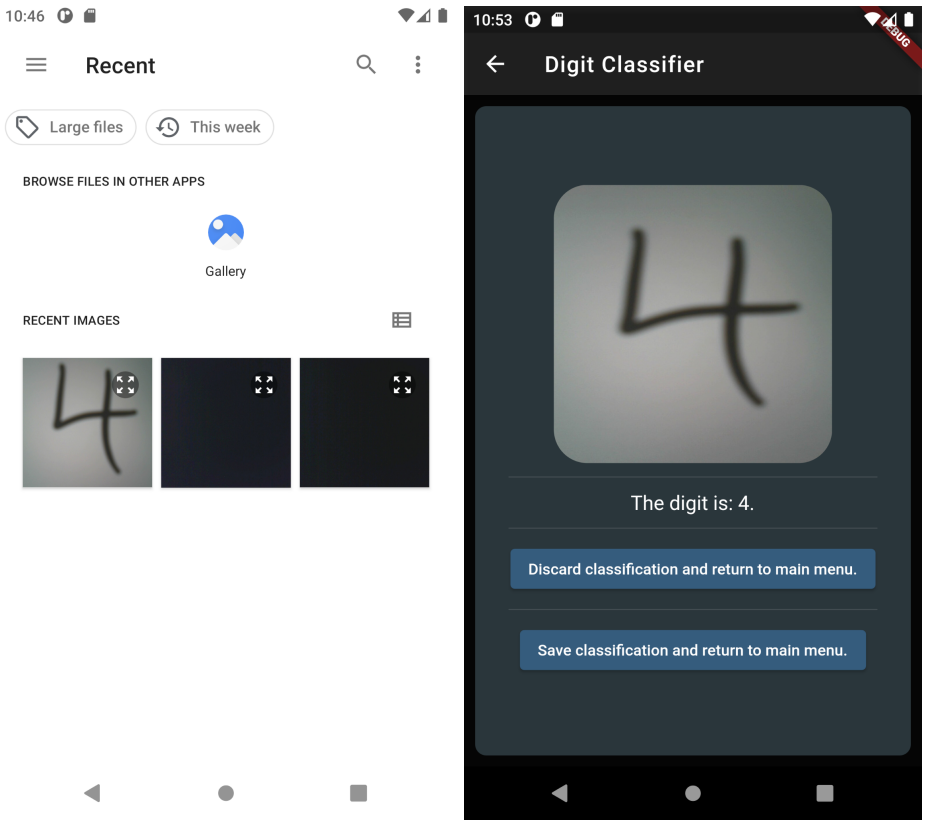
Screenshots



Main Screen: This screen demonstrates the options the user has at the main menu.

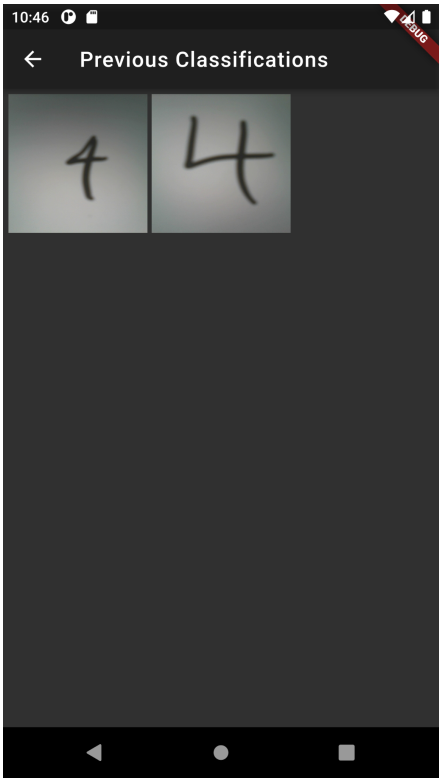


Take Photo: This screen demonstrates how a user can take a photo using their camera.

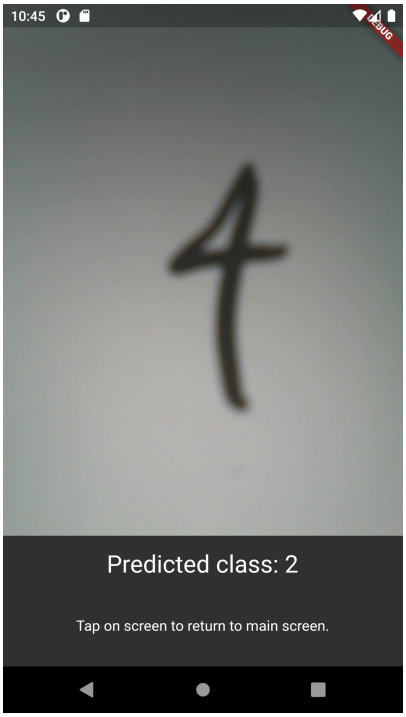


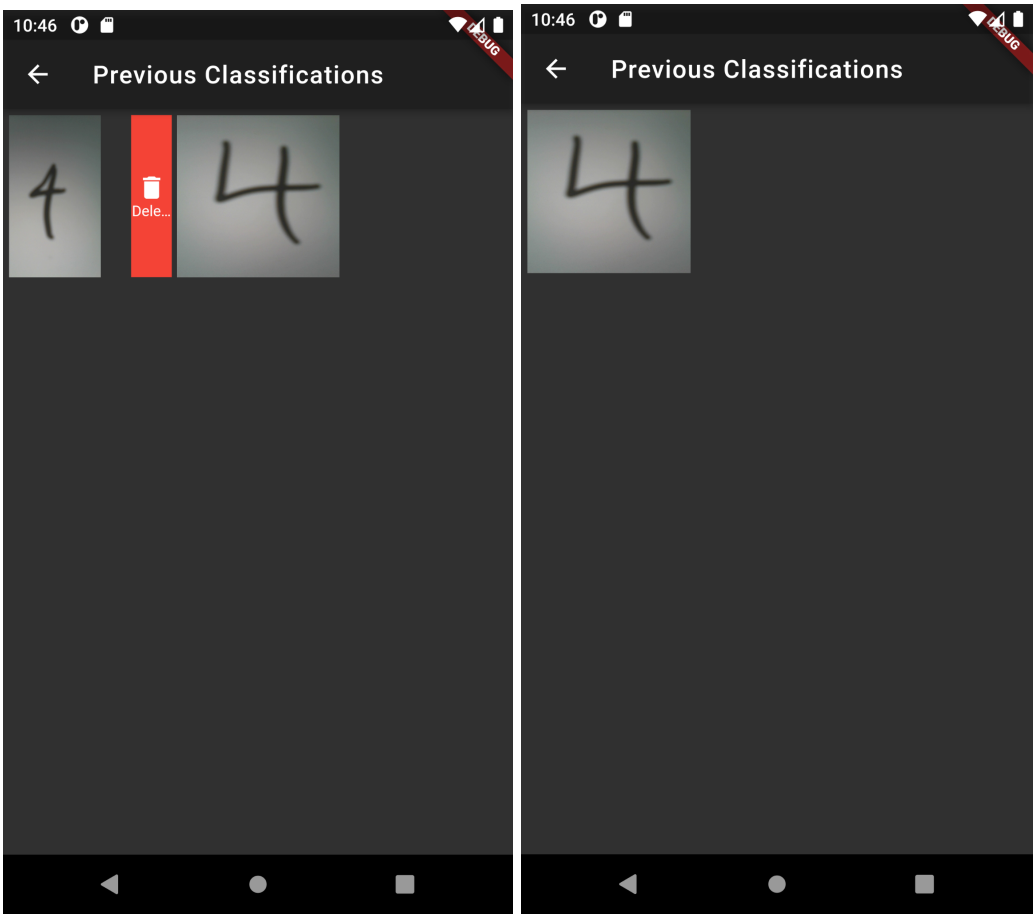
Left: **The left screen demonstrates the gallery view so that the user can select a photo from their gallery that they want to classify.**

Right: **The right screen demonstrates the results screen where it allows the user to discard or save their classification and return to the main menu so they can continue classifying if they want to.**

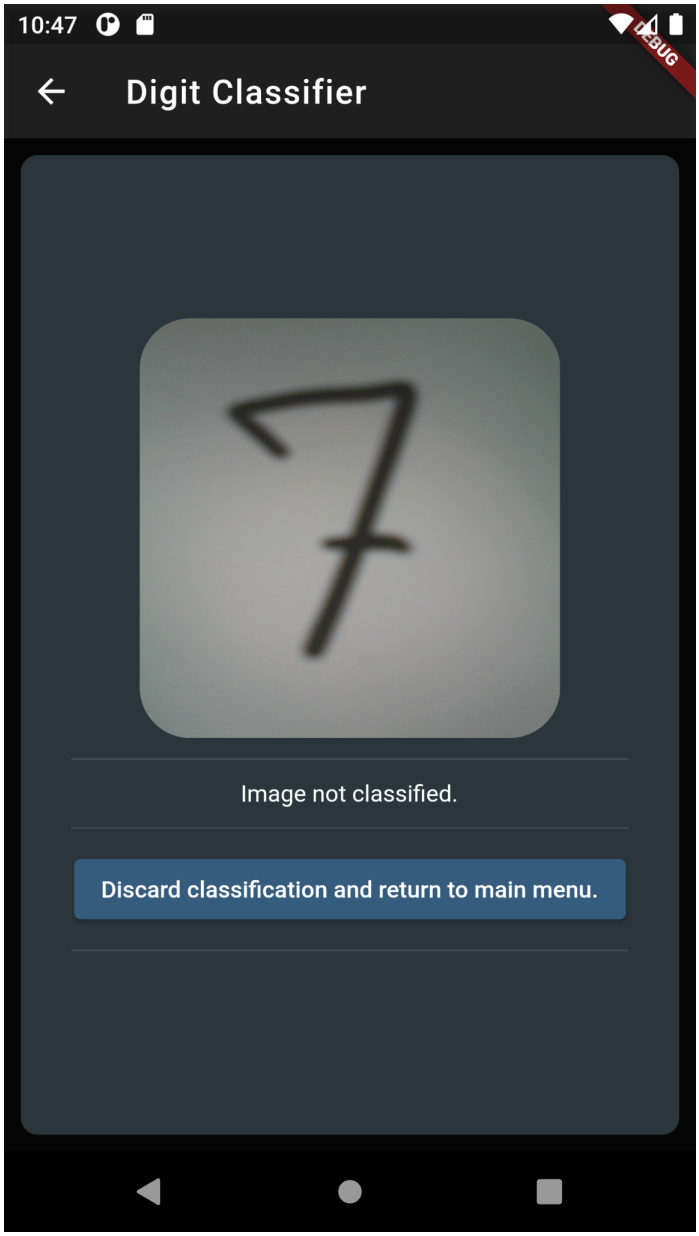


Previous Classifications Screen: **This screen displays all images that have been classified. If the user taps one of these images, it will display the image in full and tell the user what classification was predicted as shown below.**





These images demonstrate the application's ability to delete a classification. It is a slidable button.



This image demonstrates our conditional button. Since no prediction past the threshold was made, the application does not allow for saving it.

Custom Progress Indicator

While the progress indicator circles around, the images in the middle alternate back and forth between mnist dataset images. This custom progress indicator was removed from the final build since we later discovered that tflite is very fast and predictable, demonstrating 0 need for a progress indicator.

UML

Semester Project UML