App Development Journal 2

Aditya Singh, BUSN-208

Overall Goal

The original intent was to create a single iOS app that uses two Apple CoreML packages. That said, I wanted to take things a couple steps further, so I made two apps: the first one uses MNIST (single digit recognition) and YOLOv3 (most prominent object recognition), while the second one involves more UI elements and animations, but only uses MobileNetV2 (object recognition).

App #1: 1_MNIST_YOLOV3

Implementation

Using LLMs, I positioned 4 drawing canvases in a square pattern and loaded each with MNIST capabilities. Then, I instructed the app to take the most probable output from each canvas and multiplied them together to generate a final output.

CoreML Model: MNIST (Single Digit Recognition)

Purpose: Convert four independent single-digit inputs to integer values. Then, multiply the two-digit number formed by the upper two digits (e.g. 7+3=73) by the two-digit number formed by the lower two digits into a single integer value.

Input: Drawings

Output: Integers

Notes: The accuracy of individual digits ultimately depends on how close the representation of the digit is to the baseline established in the MNIST model's training data. This can lead to some inaccuracies in testing.

Implementation

I created a very simple image to text converter that takes in an image and outputs a list of the images that it detects. This has proven to be fairly inaccurate and would need further development at a much more advanced level in order to be completely usable in any real-world context, but the core functionality of YOLO is preserved.

CoreML Model: YOLOv3 (Object Detection)

Purpose: Enter an image and view an itemized list of what it contains.

Input: Image

Output: List of contents

Notes: This has been notoriously buggy on the default pictures of plants that are loaded into the native iOS simulators, so there may be issues in testing.

App #2: 2_MOBILENETV2

Implementation

I used more UI elements (splash pages, animations, etc.) to supplement an implementation of the core functionality of MobileNetV2 (object detection). Once the user follows the given instructions and inputs an object, the app will output an emoji that most closely corresponds to the object and will display a description of what it saw, a slider that rœepresents its confidence level, and a figure that represents the confidence level (e.g. 98%).

CoreML Model: MobileNetV2 (Object Recognition)

Purpose: Enter an image and receive a single output of what the object contains, as well as an emoji (from a limited set) and a level of confidence.

Input: Image

Output: Single item

Notes/Accuracy: This has been…somewhat accurate throughout my testing. It is very consistent from input to input, but continues to fail to recognize plants.

Iteration

The majority of my app was developed with the help of ChatGPT, since Cursor could not handle the integration of the CoreML models. In the spirit of full transparency, the GPT prompts that I used are included in the links below. That said, the majority of these prompts deal with error handling and the addition of other models to the app that were not originally included. For the latter, I encountered significant issues with DETR and BERT-SQuAD specifically (and any models not included in these apps in general), so please know that no substance exsits in those prompts. Further, I did not use any code directly packaged by GPT, rather I just took instructions from it and copied in larger batches of code that involved concepts that were not covered in the scope of this class.

Finally, “this” refers to errors showing up in specific screenshots that may not be visible in the shared link.

Link #1: https://chatgpt.com/share/6806a6f3-9ef4-8001-b5e6-f6023e029f4a
Link #2: https://chatgpt.com/share/6806a7bd-f0e0-8001-b131-02e6c2c8074c

To answer the questions:

What is the app Idea: The idea for the app was to generate a playground-style experience that gives me insight on how to develop an app using the models and allows the user to interact with the models at their most core forms. There was no complication or larger scheme here: this is an introduction to the Apple CoreML models for both the developer and the user.

How is AI/ML playing a role in your app?

AI/ML is the driving force behind the app – this is meant to give the user a playground-style experience within the app by letting them experiment with various features and functions that are offered by Apple and the CoreML libraries.

MNIST: Allows them to experiment by drawing out digits on the canvases and looking at the results. This comes with varying levels of accuracy, as expected, but is ultimately a good learning experience on how the way models are trained can affect the accuracy of the model.

YOLO/MobileNet: This allows the users to compare and contrast two different object recognition models and examine how closely related they are. They both have wildly different accuracy levels which could ultimately influence how they respond to the same input.

What were the considerations when developing the user interface?

This was fairly simple – basic and intuitive. Given that the driving purpose of the application is to offer a playground-style experience, I didn't want this to be super complicated. Rather, I wanted the users to be able to access the core functionality, in similar fashion to the app presented to us, as quickly as possible.

Are there similar apps out there? Discuss how yours stand in comparison?

There are not similar apps out there that are published, but that can be changed. The app in itself is fairly simplistic: ultimately, the goal here was to allow the users to get to know the functionality and variability of the models as quickly and intuitively as possible. It's also wrapped in a larger scheme of encouraging them to not trust the models blindly: though the model believes that it is confident about something, it may not necessarily be.