Object Scanner (10 pts.)

Lab Objectives:

- Use the CameraX API
- Use ML Kit
- Create an app that uses machine learning to identify objects in the camera view

Background

In a previous app, we used the original Android Camera API. In this lab, we will use the newer CameraX API to add a camera preview to our app, and to enable analysis of the images streamed from the camera. Additionally, we will use Google's ML Kit to classify the images. We will do this using Google's default image classifier, without needing to know anything about machine learning.

Steps

1. Create a new project with an empty Views Activity. Use a min SDK of at least API level 21.

2. Add dependencies for CameraX, as described here: https://developer.android.com/training/camerax/architecture#dependencies

   If you get an error about duplicate classes, try adding this dependency:
   implementation(platform("org.jetbrains.kotlin:kotlin-bom:1.8.0"))

   You may need to set your compile SDK to at least API 34. If you don't see API 34 as an option in the Module settings, download it using the SDK manager.

3. Add code to request the CAMERA permission, similar to how you requested location permissions in a previous lab.

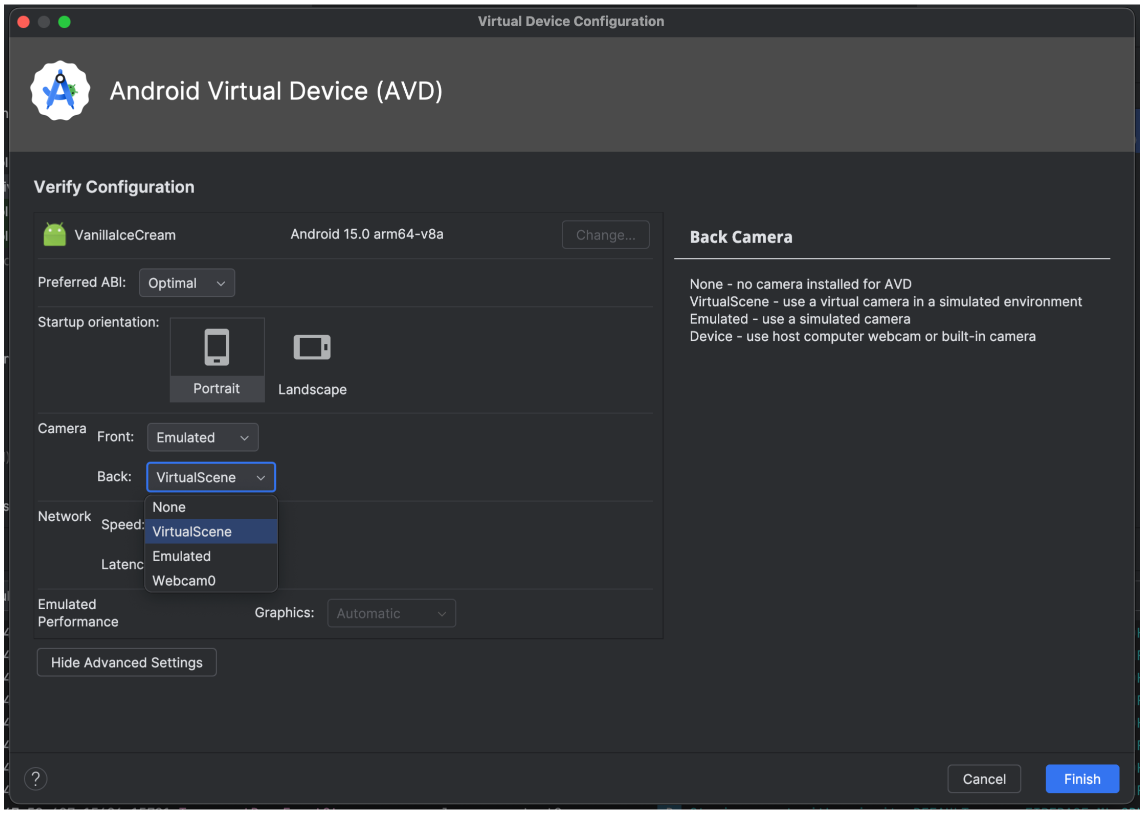
4. Delete the default TextView in activity_main.xml. Add and bind a camera "preview," as described here: https://developer.android.com/training/camerax/preview. You may skip the step for CameraXConfig.Provider.

5. Add the image analysis "use case," as described here: https://developer.android.com/training/camerax/analyze. Use the non-blocking backpressure strategy. You should use an Executor that will execute tasks (the image analysis) on a background/worker thread; check the course module on background tasks and threading if needed.

6. Now let's use machine learning to classify the objects in the image in real-time. Use Google's ML Kit to label the image, as described here: https://developers.google.com/ml-kit/vision/image-labeling/android. Hook into your analyze() callback from the previous step.

Hint: make sure you call close() on the ImageProxy in the success/failure listeners of ImageLabeler.process(), NOT in analyze().

Hint: if you're using the emulator and the camera is just showing random colors and you want a more interesting image to analyze, go to Tools -> Device Manager and edit your virtual device. Click Show Advanced Settings and enable VirtualScene for the back camera.



VanillaIceCream Android 15.0 arm64-v8a

Preferred ABI: Optimal

Startup orientation: Portrait

Camera Front: Emulated

Back: VirtualScene

Network Speed:

Latency:

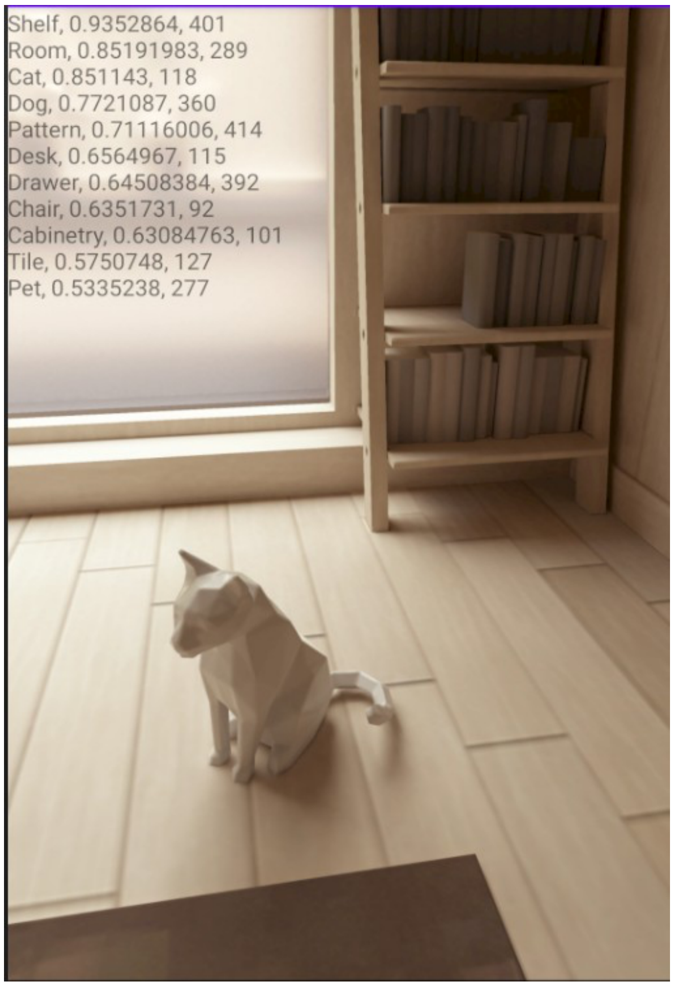
Emulated Performance Graphics: Automatic

Back Camera

None - no camera installed for AVD
VirtualScene - use a virtual camera in a simulated environment
Emulated - use a simulated camera
Device - use host computer webcam or built-in camera

7. If it works, a list of ImageLabels will be returned in the success listener. These labels are the results of the image classifier run by ImageLabeler. The text of the label is how the classifier has categorized the image. The confidence score is how certain the classifier is that the label is correct. The index is the logical identifier of the label.

   Display the text, confidence scores, and indices for the labels in a TextView overlaying the camera preview. A sample image with labels is shown below.



```
Shelf, 0.9352864, 401
Room, 0.85191983, 289
Cat, 0.851143, 118
Dog, 0.7721087, 360
Pattern, 0.71116006, 414
Desk, 0.6564967, 115
Drawer, 0.64508384, 392
Chair, 0.6351731, 92
Cabinetry, 0.63084763, 101
Tile, 0.5750748, 127
Pet, 0.5335238, 277
```

What to Turn in:

- Demonstrate your working app to a TA.
  - Camera preview (4 pts.)
  - Analysis on a background thread (3 pts.)
  - Accurate-ish labels displayed on the screen (3 pts.)

Demonstrate your working application to one of the TAs during lab time or TA student hours.