**CMU 04-801: Advanced Mobile Application Development**
**Week 7: ML Kit**

## Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

Machine Learning process:

1. Define a problem
2. Collect and prepare data
3. Develop a model
4. Train model
5. Evaluate and tune the model
6. Deploy the model
7. Prediction/Inference

## Machine Learning on Mobile

https://developer.android.com/ml

Can you think of some interesting applications of ML on the mobile platform?

- Computer vision
    - Image recognition
    - Object detection
    - Pose estimation
    - Facial recognition
    - Text recognition
        - Optical Character Recognition (OCR)
- Augmented reality
- Natural language processing
    - Text classification
    - Translation
    - Auto complete/smart reply
- Audio
    - Audio recognition
    - Speech recognition

On-device

- Low latency
    - Snapchat filters
- Low/no connectivity
- Security/Privacy
- Reduced cost

Backend server

- More powerful
- Higher level of accuracy
- More storage (more samples provided)

Although it's still early, both Apple and Google provide libraries and tools to help mobile developers integrate ML into mobile apps without needing to be a ML expert.

iOS

- Core ML
  - a unified representation for all machine learning models that are ready to deploy to iOS.
- Create ML
  - a high-level Core ML model training tool that makes it easy to get started with ML
- Turi Create
  - A Python-based, high-level training framework maintained by Apple
  - more flexible than Create ML

Android

- ML Kit
  - offers pre-trained models for common ML tasks
  - the ability to create hybrid systems where some server-side and on-device models are usable via the same API
  - ML Kit can be used on iOS as wellß

Cross-platform tools

- TensorFlow Lite
  - provides tools to convert and run TensorFlow models on mobile, embedded, and IoT devices
- PyTorch Mobile
  - A framework for helping mobile developers and machine learning engineers embed PyTorch ML models on-device
- Fritz AI
  - provides models compatible with the fastest runtime on every platform
  - cross-platform
- Skafos
  - a tool for deploying machine learning models to mobile apps and managing the same models in a production environment
  - integrates with any of the major cloud providers
- Caffe2
  - a lightweight, modular, scalable deep learning framework developed by Facebook
  - built for production use cases and mobile development
  - offers developers greater flexibility for building high-performance products
- OpenCV
  - a collection of programming functions for real-time computer vision and machine learning
  - supports the deep learning frameworks TensorFlow and PyTorch

**Google ML Kit**

https://firebase.google.com/docs/ml-kit/

ML Kit is a mobile SDK that makes common machine learning tasks easy and accessible to developers.

ML Kit comes with a set of ready-to-use APIs for common mobile use cases

- no need to have deep knowledge of neural networks or model optimization to get started

Custom models

- includes APIs that help you use your custom TensorFlow Lite models in your mobile apps

On-device or in the cloud

- on-device APIs can process your data quickly and work even when there's no network connection
- cloud-based APIs using Google Cloud Platform's machine learning technology provide a higher level of accuracy

Cross platform

- Android and iOS

Note that ML kit is still listed as in beta.

Ready to Use APIs

ML Kit comes with a set of ready-to-use APIs for common mobile use cases
Some are available on-device, in the cloud, or both https://firebase.google.com/docs/ml-kit/#what_features_are_available_on_device_or_in_the_cloud
Vision

- text recognition https://firebase.google.com/docs/ml-kit/recognize-text
- face detection https://firebase.google.com/docs/ml-kit/detect-faces
- barcode scanning https://firebase.google.com/docs/ml-kit/read-barcodes
- image labeling https://firebase.google.com/docs/ml-kit/label-images
- object detection and tracking https://firebase.google.com/docs/ml-kit/object-detection
- landmark recognition https://firebase.google.com/docs/ml-kit/recognize-landmarks

Natural Language

- language identification https://firebase.google.com/docs/ml-kit/identify-languages
- translation https://firebase.google.com/docs/ml-kit/translation
- smart reply https://firebase.google.com/docs/ml-kit/generate-smart-replies

**Quickstart demo app**

Google provides a quickstart demo app for ML Kit that is part of their Firebase quickstart repo.
https://github.com/firebase/quickstart-android/tree/master/mlkit
Even if you won't be using it with Firebase, you need to add the Firebase setup for it to compile and run.
Add Firebase to your Android App https://firebase.google.com/docs/android/setup
Register the app with a Firebase project by adding the package to a Firebase project.
`com.google.firebase.samples.apps.mlkit`
Then download the google-services.json file from that Firebase project and add it to the app directory in your project.
Resync your project and it should build without errors.
File | Sync Project with Gradle Files
Note that there are gradle dependencies for each ML detector.

EntryChoiceActivity.kt lets the user choose between Java and Kotlin demos.

ChooserActivity.java lets you choose between Live Preview using the live camera and Still Image where you can take or use an image from an album.
It also handles the required permissions.

LivePreviewActivity

UI and camera setup.
For each option the class extends VisionProcessorBase.java

VisionProcessorBase implements VisionImageProcessor which is an interface that allows us to process images with different ML Kit detectors.

process() is an overloaded method that processes the images
Any class that extends the VisionProcessorBase class will need to implement these required methods from the VisionImageProcessor interface:

- detectInImage()
    - takes a FirebaseVisionImage object which is required by the detectorsß
    - returns the result of whatever detection is being performed.
- onSuccess() handles the results and displays them on the screen using a GraphicOverlay.
    - A Graphic overlay is UI layer that allows us to display things on the camera feed
- onFailure() handles errors

Face Detection
https://firebase.google.com/docs/ml-kit/face-detection-concepts
Face detection is the process of automatically locating human faces in visual media (digital images or video). A face that is detected is reported at a position with an associated size and orientation.

- **Face tracking** extends face detection to video sequences. Any face appearing in a video for any length of time can be tracked. That is, faces that are detected in consecutive video frames can be identified as being the same person.
    - Note that this is not a form of face recognition; this mechanism just makes inferences based on the position and motion of the faces in a video sequence.
- A **landmark** is a point of interest within a face.
    - The left eye, right eye, and base of the nose are all examples of landmarks.
    - ML Kit provides the ability to find landmarks on a detected face.
- A **contour** is a set of points that follow the shape of a facial feature.
    - ML Kit provides the ability to find the contours of a face.
- **Classification** determines whether a certain facial characteristic is present.
    - ML Kit currently supports two classifications: **eyes open** and **smiling**.

Creating an instance of the FirebaseVisionFaceDetector class requires settings with a FirebaseVisionFaceDetectorOptions object.
Settings:

- Performance mode
    - FAST (default): favors speed
    - ACCURATE: favors accuracy
- Detect landmarks: Whether to attempt to identify facial "landmarks": eyes, ears, nose, cheeks, mouth, and so on.
    - NO_LANDMARKS (default):
    - ALL_LANDMARKS
- Detect contours: Whether to detect the contours of facial features. Contours are detected for only the most prominent face in an image.
    - NO_CONTOURS (default)
    - ALL_CONTOURS
- Classify faces: Whether or not to classify faces into categories such as "smiling", and "eyes open"
    - NO_CLASSIFICATIONS (default)
    - ALL_CLASSIFICATIONS

FaceDetectionProcessor:
Create a FirebaseVisionFaceDetectorOptions options object.

```
FirebaseVisionFaceDetectorOptions options =
        new FirebaseVisionFaceDetectorOptions.Builder()
      .setClassificationMode(FirebaseVisionFaceDetectorOptions.ALL_CLASSIFICATIONS)
                .setLandmarkMode(FirebaseVisionFaceDetectorOptions.ALL_LANDMARKS)
                .build();
```

Create a FirebaseVisionFaceDetector object

```
FirebaseVision.getInstance().getVisionFaceDetector(options);
```

detectInImage() takes an image from the camera to detect faces.
FirebaseVisionFaceDetector returns FirebaseVisionFace objects which represent a face

- A List is returned because it could detect multiple faces
- The List is passed to onSuccess()

onSuccess()

- Clears the graphic overlay
- loops through the list of FirebaseVisionFace objects
- makes a face graphic for each face and add it to the graphic overlay

In the FaceGraphic class the draw() and drawLandmarkPosition() methods draw the circles on the landmarks and the clown nose bitmap on the nose landmark.

Face Contour
FaceContourDetectorProcessor:
Uses the same FirebaseVisionFaceDetector class and FirebaseVisionFaceDetectorOptions but uses the contour mode of "ALL_CONTOURS".
The rest stays the same but the FaceContourGraphic class draws the contour of the face.

Text Detection
https://firebase.google.com/docs/ml-kit/android/recognize-text
The ML Kit API recognizes text in any Latin-based language on device and a broader range in the cloud.
FirebaseVisionTextRecognizer performs optical character recognition(OCR) on an input image.

TextRecognitionProcessor:
Create a FirebaseVisionTextRecognizer object

```
FirebaseVision.getInstance().getOnDeviceTextRecognizer();
```

detectInImage() takes an image from the camera and feeds it into FirebaseVisionTextRecognizer

The onSuccess() method takes the results and displays them on the screen using the graphic overlay

- Clears the graphic overlay
- Gets the blocks from the results
- Blocks contain a list of lines
- Lines contain a list of elements
- For each element a new TextGraphic is created for rendering and added to the graphic overlay

Image Label Detection

https://firebase.google.com/docs/ml-kit/label-images
The ML Kit Image Labeling API gives you a list of labels that were recognized that cover the most commonly-found concepts in photos: people, things, places, activities, etc. On-device detection includes 400+ labels and detection in the cloud 10,000+ labels are available.

ImageLabelingProcessor:
Create an instance of FirebaseVisionImageLabeler.
```
FirebaseVision.getInstance().getOnDeviceImageLabeler();
```

Can also take an instance of FirebaseVisionLabelDetectorOptions that specifies the confidence threshold.

detectInImage() takes an image from the camera and feeds it into FirebaseVisionImageLabeler to process image labels.

FirebaseVisionImageLabeler returns FirebaseVisionLabel objects

- getEntityId() returns the entity id
- getConfidence returns the overall confidence (0.0 to 1.0 float)
    - By default, ML Kit only returns labels with a confidence threshold of 0.5 or higher

onSuccess() is passed a List of FirebaseVisionLabel objects

- Clears the graphic overlay
- LabelGraphic loops through the labels and draws the text on the graphic overlay

StillImageActivity
The options in StillImageActivity all use cloud-based processing.

Text Detection
CloudTextRecognitionProcessor :
The only difference is when we get an instance of FirebaseVisionTextRecognizer we request the cloud recognizer instead of the onDevice recognizer.
```
FirebaseVision.getInstance().getCloudTextRecognizer();
```

CloudDocumentTextRecognitionProcessor:
FirebaseVisionDocumentTextRecognizer is the cloud-based document text recognizer used for performing optical character recognition(OCR) on an input image which can be better for dense text in documents.

processImage() detects text from supplied document image and returns a FirebaseVisionDocumentText object.
onSuccess() parses the FirebaseVisionDocumentText object.

- getText() returns the text
- getBlocks() returns the blocks that you can loop through to parse the paragraphs, words,

Label Detection
CloudImageLabelingProcessor:
`The only difference` is when we get an instance of `FirebaseVisionImageLabeler`
we request the cloud recognizer instead of the onDevice recognizer and pass it options.
```
FirebaseVision.getInstance().getCloudImageLabeler(optionsBuilder.build());
```