Face Recognition
Face Recognition

- Answers the question:
  
  Who is this person in this picture?
  
  Example answer: John Smith

- Compares unknown face to database of faces with known identity

- Neural networks/deep learning now faster, more accurate comparison
FindFace App: Stalking on Steroids?

- See stranger you like? Take a picture
- App searches 1 billion pictures using neural networks < 1 second
- Finds person’s identity, link on VK (Russian Facebook)
  - You can send friend Request
- ~ 70% accurate!
- Can also upload picture of celebrity you like
- Finds 10 people on Facebook who look similar, can send friend request
FindFace App

- Also used in law enforcement
  - Police identify criminals on watchlist

Face Detection
Mobile Vision API
https://developers.google.com/vision/

- **Face Detection**: Are there [any] faces in this picture?
- **How?** Locate face in photos and video and
  - **Facial landmarks**: Eyes, nose and mouth
  - **State of facial features**: Eyes open? Smiling?
Face Detection: Google Mobile Vision API

Ref: https://developers.google.com/vision/face-detection-concepts

- Detects faces:
  - reported at a position, with size and orientation
  - Can be searched for landmarks (e.g. eyes and nose)

**Orientation**

<table>
<thead>
<tr>
<th>Euler Y angle</th>
<th>Detectable landmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; -36 degrees</td>
<td>left eye, left mouth, left ear, nose base, left cheek</td>
</tr>
<tr>
<td>-36 degrees to -12 degrees</td>
<td>left mouth, nose base, bottom mouth, right eye, left eye, left cheek, left ear tip</td>
</tr>
<tr>
<td>-12 degrees to 12 degrees</td>
<td>right eye, left eye, nose base, left cheek, right cheek, left mouth, right mouth, bottom mouth</td>
</tr>
<tr>
<td>12 degrees to 36 degrees</td>
<td>right mouth, nose base, bottom mouth, left eye, right eye, right cheek, right ear tip</td>
</tr>
<tr>
<td>&gt; 36 degrees</td>
<td>right eye, right mouth, right ear, nose base, right cheek</td>
</tr>
</tbody>
</table>
Google Mobile Vision API

- Mobile Vision API also does:
  - **Face tracking**: detects faces in consecutive video frames
  - **Classification**: Eyes open? Face smiling?

- Classification:
  - Determines whether a certain facial characteristic is present
  - API currently supports 2 classifications: eye open, smiling
  - Results expressed as a confidence that a facial characteristic is present
    - E.g. > 0.7 confidence means likely person is smiling

- Mobile vision API does face **detection** but NOT **recognition**
Face Detection

- **Face detection**: Special case of object-class detection

- **Object-class detection task**: find locations and sizes of all objects in an image that belong to a given class.
  - E.g: bottles, cups, pedestrians, and cars

- **Object matching**: Objects in picture compared to objects in database of labelled pictures
Mobile Vision API: Other Functionality

- Barcode scanner
- Recognize text
Face Detection Using Google’s Mobile Vision API
Getting Started with Mobile Vision Samples

- Get **Android Play Services SDK** level 26 or greater
- Download mobile vision samples from github

Sample code for the Android Mobile Vision API. [https://developers.google.com/vision/](https://developers.google.com/vision/)

- Branch: **master**
- New pull request

Files:
- `.google`
- `visionSamples`
- `.gitignore`
- `LICENSE`
- `README.md`
Creating the Face Detector

Ref: https://developers.google.com/vision/android/detect-faces-tutorial

- In app’s **onCreate** method, create face detector

```java
FaceDetector detector = new FaceDetector.Builder(context)
    .setTrackingEnabled(false) // Don't track points
    .setLandmarkType(FaceDetector.ALL_LANDMARKS) // Detect all landmarks
    .build();
```

- **detector** is base class for implementing specific detectors. E.g. face detector, bar code detector
- Tracking finds same points in multiple frames (continuous)
- Detection works best in single images when **trackingEnabled** is false
Detecting Faces and Facial Landmarks

- Create Frame (image data, dimensions) instance from bitmap supplied
  
  ```java
  Frame frame = new Frame.Builder().setBitmap(bitmap).build();
  ```

- Call detector synchronously with frame to detect faces
  
  ```java
  SparseArray<Face> faces = detector.detect(frame);
  ```

- Detector takes `Frame` as input, outputs array of `Faces`
- `Face` is a single detected human face in image or video
- Iterate over array of faces, landmarks for each face, and draw the result based on each landmark position

```java
for (int i = 0; i < faces.size(); ++i) {
    Face face = faces.valueAt(i);
    for (Landmark landmark : face.getLandmarks()) {
        int cx = (int) (landmark.getPosition().x * scale);
        int cy = (int) (landmark.getPosition().y * scale);
        canvas.drawCircle(cx, cy, 10, paint);
    }
}
```
Other Stuff

- To count faces, call `faces.size()`

```java
TextView faceCountView = (TextView) findViewById(R.id.face_count);
faceCountView.setText(faces.size() + " faces detected");
```

- Querying Face detector’s status

```java
if (!detector.isOperational()) {
    // ...
}
```

- Releasing Face detector (frees up resources)

```java
detector.release();
```
Detect & Track Multiple Faces in Video

- Can also track multiple faces in image sequences/video, draw rectangle round each one
Face Interpretation
Visage Face Interpretation Engine

- Real-time face interpretation engine for smartphones
  - Tracking user’s 3D head orientation + facial expression

- Facial expression?
  - angry, disgust, fear, happy, neutral, sad, surprise
  - Can be used in Mood Profiler app

Facial Expression Inference

- Active appearance model
  - Describes 2D image as triangular mesh of landmark points
- 7 expression classes: angry, disgust, fear, happy, neutral, sad, surprise
- Extract triangle shape, texture features
- Classify features using Machine learning
Classification Accuracy

<table>
<thead>
<tr>
<th>Expressions</th>
<th>Anger</th>
<th>Disgust</th>
<th>Fear</th>
<th>Happy</th>
<th>Neutral</th>
<th>Sadness</th>
<th>Surprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy(%)</td>
<td>82.16</td>
<td>79.68</td>
<td>83.57</td>
<td>90.30</td>
<td>89.93</td>
<td>73.24</td>
<td>87.52</td>
</tr>
</tbody>
</table>
Skipped Android Nerd Ranch
CriminalIntent Chapters
Chapter 9: Displaying Lists with RecyclerView

- RecyclerView facilitates view of large dataset

- E.g. Allows crimes in CriminalIntent to be listed
Chapter 11: Using ViewPager

- ViewPager allows users to swipe between screens (e.g., Tinder?)
- E.g., Users swipe between Crimes in CriminalIntent
Chapter 12: Dialogs

- Dialogs present users with a choice or important information
- E.g. DatePicker allows users pick date
- Allows users to pick a date on which a crime occurred in CriminalIntent
Chapter 13: The Toolbar

- Toolbar includes actions user can take
- In CriminalIntent, menu items for adding crime, navigate up the screen hierarchy
Android Nerd Ranch Ch 14
SQLite Databases
Background on Databases

- Relational DataBase Management System (RDBMS)
  - Introduced by E. F. Codd (Turing Award Winner)

- Relational Database
  - data stored in tables
  - relationships among data stored in tables
  - data can be accessed and viewed in different ways
Example Wines Database

- **Relational Data:** Data in different tables can be related

Ref: Web Database Applications with PHP and MySQL, 2nd Edition, by Hugh E. Williams, David Lane
Keys

- Each table has a key
- **Key:** column used to uniquely identify each row
SQL and Databases

- **SQL**: language used to manipulate information in a Relational Database Management System (RDBMS)

- **SQL Commands:**
  - **CREATE TABLE** - creates new database table
  - **ALTER TABLE** - alters a database table
  - **DROP TABLE** - deletes a database table
  - **SELECT** - get data from a database table
  - **UPDATE** - change data in a database table
  - **DELETE** - remove data from a database table
  - **INSERT INTO** - insert new data in a database table
CriminalIntent Database

- **SQLite**: open source relational database
- SQLite implements most, but not all of SQL
  - [http://www.sqlite.org/](http://www.sqlite.org/)
- Android includes SQLite database
- **Goal**: Store crimes in CriminalIntent in SQLite database
- First step, define database table of crimes

<table>
<thead>
<tr>
<th>_id</th>
<th>uuid</th>
<th>title</th>
<th>date</th>
<th>solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13090636733242</td>
<td>Stolen yogurt</td>
<td>13090636733242</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>13090732131909</td>
<td>Dirty sink</td>
<td>13090732131909</td>
<td>1</td>
</tr>
</tbody>
</table>
CriminalIntent Database

- Create `CrimeDbSchema` class to store crime database
- Define columns of the Crimes database table

```java
public class CrimeDbSchema {
    public static final class CrimeTable {
        public static final String NAME = "crimes"; // Name of Table

        public static final class Cols {
            public static final String UUID = "uuid"; // UUID
            public static final String TITLE = "title"; // Title
            public static final String DATE = "date"; // Date
            public static final String SOLVED = "solved"; // Solved
        }
    }
}
```

<table>
<thead>
<tr>
<th>_id</th>
<th>uuid</th>
<th>title</th>
<th>date</th>
<th>solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13090636733242</td>
<td>Stolen yogurt</td>
<td>13090636733242</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>13090732131909</td>
<td>Dirty sink</td>
<td>13090732131909</td>
<td>1</td>
</tr>
</tbody>
</table>
SQLiteOpenHelper

- **SQLiteOpenHelper** class used for database creation, opening and updating
- In **CriminalIntent**, create subclass of **SQLiteOpenHelper** called **CrimeBaseHelper**

```java
public class CrimeBaseHelper extends SQLiteOpenHelper {
    private static final int VERSION = 1;
    private static final String DATABASE_NAME = "crimeBase.db";

    public CrimeBaseHelper(Context context) {
        super(context, DATABASE_NAME, null, VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```

- Used to create the database (to store Crimes)
- Called the first time database is created
Use CrimeBaseHelper to open SQLite Database

```java
public class CrimeLab {
    private static CrimeLab sCrimeLab;

    private List<Crime> mCrimes;
    private Context mContext;
    private SQLiteDatabase mDatabase;

    ...

    private CrimeLab(Context context) {
        mContext = context.getApplicationContext();
        mDatabase = new CrimeBaseHelper(mContext)
                .getWritableDatabase();
        mCrimes = new ArrayList<>();
    }

    ...
}
```

- Store instance of context in variable. Will need it later
- Opens new writeable Database
Create CrimeTable in onCreate( )

```java
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("create table " + CrimeTable.NAME + "(" +
        "_id integer primary key autoincrement, " +
        CrimeTable.Cols.UUID + ", " +
        CrimeTable.Cols.TITLE + ", " +
        CrimeTable.Cols.DATE + ", " +
        CrimeTable.Cols.SOLVED + 
    ")" );
}
```

onCreate called first time database is created

Create CrimeTable in our new Crimes Database
Writing Crimes to Database using ContentValues

- In Android, writing to databases is done using class **ContentValues**
- **ContentValues** is key-value pair
- Create method to create **ContentValues** instance from a Crime

```java
public getCrime(UUID id) {
    return null;
}

private static ContentValues getContentValues(Crime crime) {
    ContentValues values = new ContentValues();
    values.put(CrimeTable.Cols.UUID, crime.getUuid().toString());
    values.put(CrimeTable.Cols.TITLE, crime.getTitle());
    values.put(CrimeTable.Cols.DATE, crime.getDate().getTime());
    values.put(CrimeTable.Cols.SOLVED, crime.isSolved() ? 1 : 0);
    return values;
}
```
Quiz 2
Quiz 2

- Quiz in class next Monday (before class Mon, 1/30)
- Short answer questions
- Try to focus on understanding, not memorization
- Covers:
  - Lecture slides for lectures 5-8
  - 1 code example from books
    - **ANR example**: geoQuiz, starting CheatActivity (Ch 5)
References

- Google Mobile Vision API, https://developers.google.com/vision/
- Busy Coder’s guide to Android version 6.3
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014