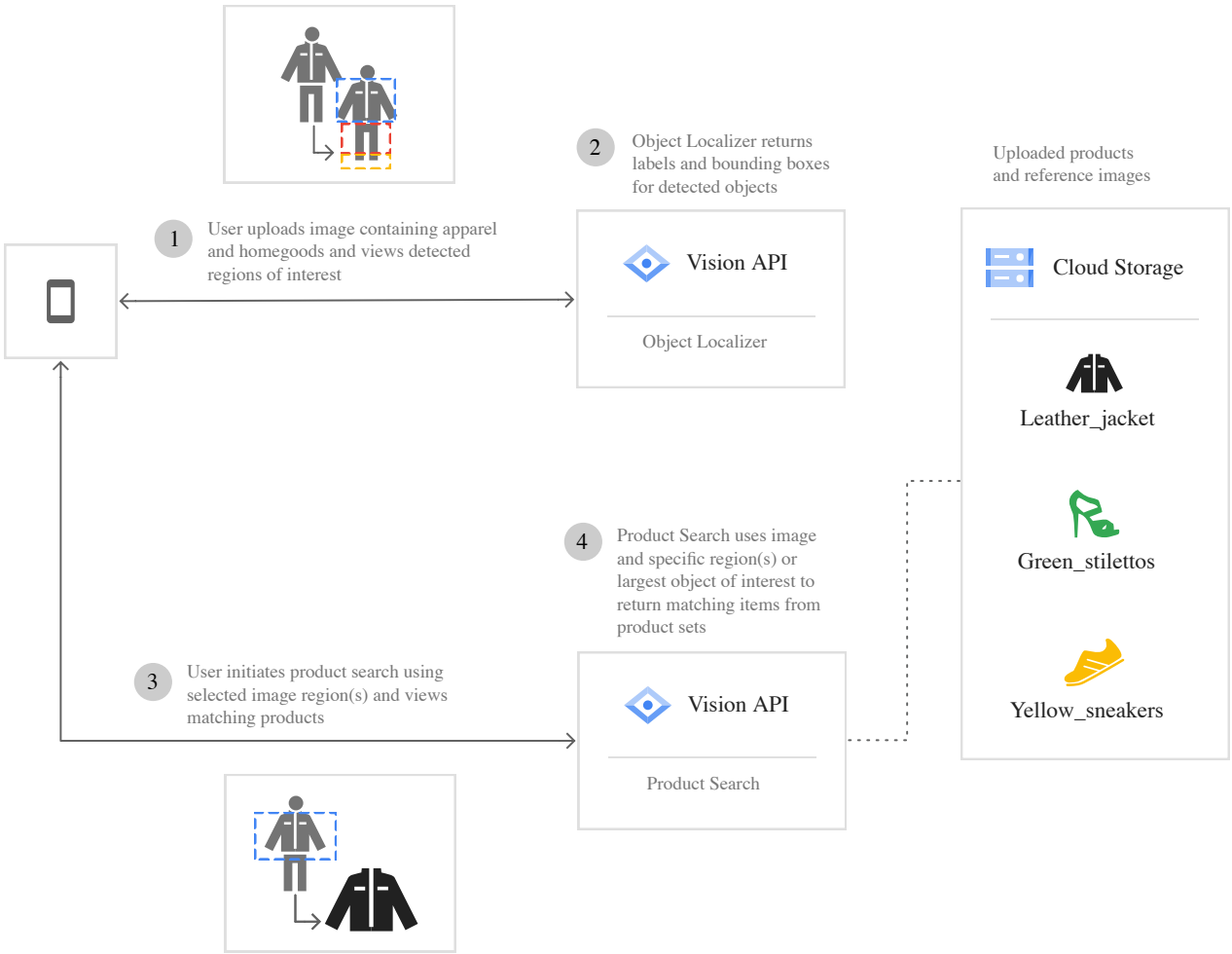
Google Cloud Vision API Document

Description:

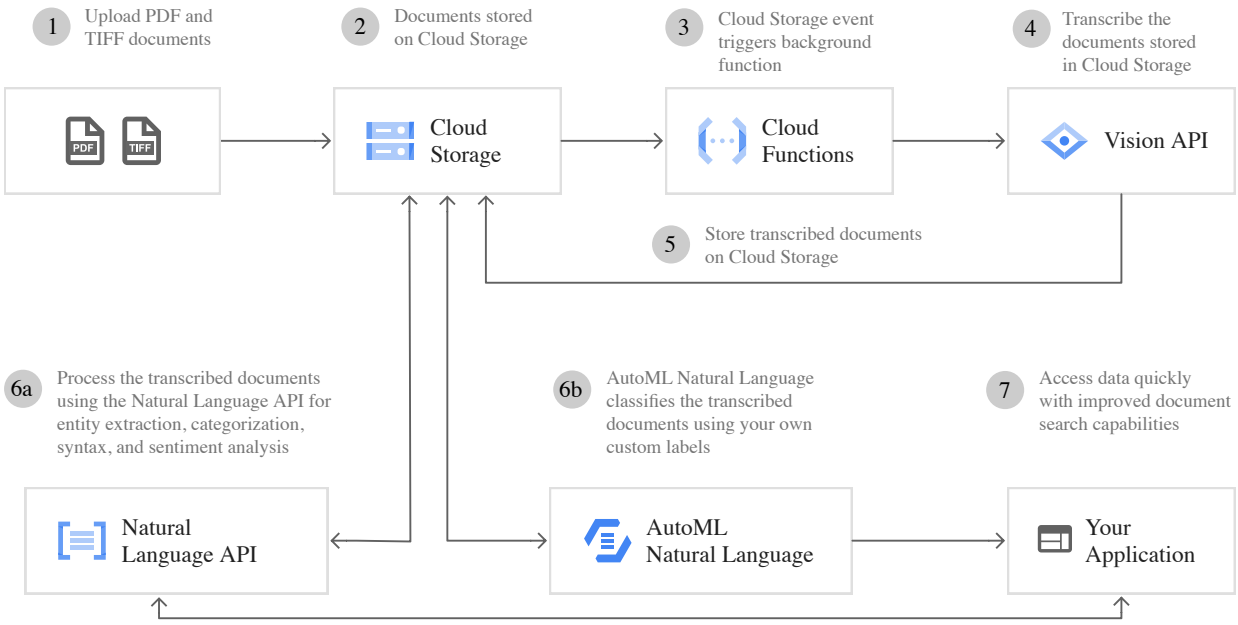
Vision API offers powerful **pre-trained** machine learning models through **REST** and **RPC** APIs. Assign labels to images and quickly classify them into millions of predefined categories. Detect objects, read printed and handwritten text, and build valuable metadata into your image catalog.

Use cases:

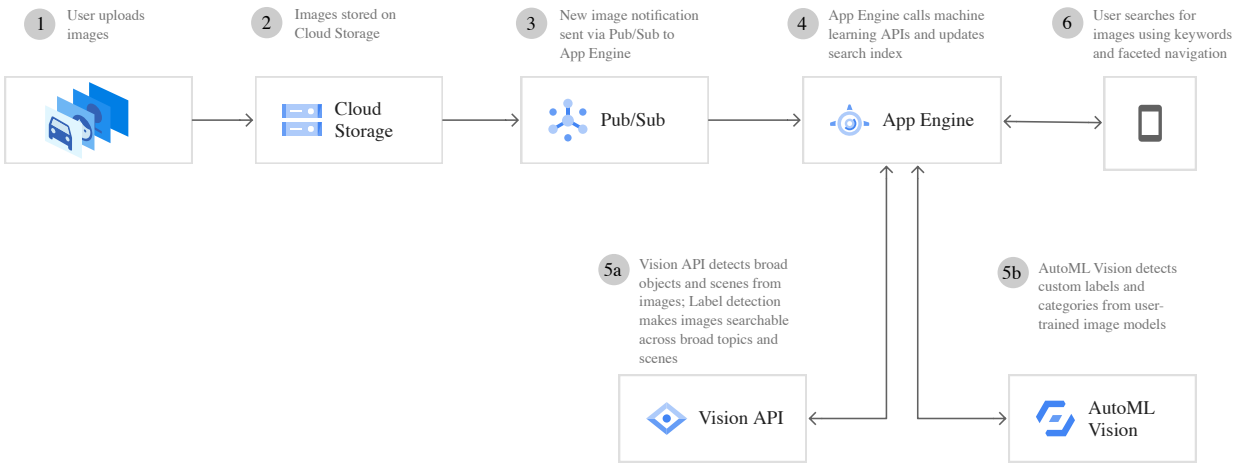
- **Product search**
  - Find products of interest within images and visually search product catalogs using Vision API.



- **Document classification**
  - Access information efficiently by using the Vision and Natural Language APIs to classify, extract, and enrich documents.



- **Image search**
  - Use Vision API and AutoML Vision to make images searchable across broad topics and scenes, including custom categories.



Personal thought:

- **Benefit**: Since this API uses pre-trained ML models, it is pretty **easy to use** for *individual* or *small business*.
- **Drawbacks**: Since the ML models is pre-trained, users **cannot** use their own model or modify the model that Vision API uses. So, the detection accuracy for some situation is **low**.
  - But there are other Google products provides this feature, such as: Vertex AI (Build, deploy, and scale machine learning (ML) models faster, with fully managed ML tools for any use case).
  - Google even suggests the users for Vision API migrate to Vertex AI.

Import Vision API Library

Vision API supports **multiple platforms**, such as: Go, Java, Node.js, Python, C#, PHP, Ruby.

- Also support REST by encoding image into **Base64** ASCII string.

Since we will use this API in an **Android** app, we need to use the official library for **Java**, and use **Gradle** as our build system.

In the `build.gradle(:app)` file:

Import the official **Vision API library**.

```
implementation 'com.google.apis:google-api-services-vision:v1-rev20220915-2.0.0'
```

Import a **HTTP Client library** so we can send request.

```
implementation 'com.google.api-client:google-api-client-android:2.0.1'
```

Import a **Gson/Json** library to handle Vision API return.

```
implementation 'com.google.http-client:google-http-client-gson:1.42.3'
```

For newer version Android system: (Prevent duplicate dependency files)

```
packagingOptions {
    exclude 'META-INF/DEPENDENCIES.txt'
    exclude 'META-INF/DEPENDENCIES'
}
```

How to use Vision API

1. Create a new Vision builder

   ```
   //Use Vision Builder to create a new builder and pass http client, and Gson factory to the builder
   Vision.Builder visionBuilder = new Vision.Builder(new NetHttpTransport(), new GsonFactory(), null);
   ```

2. Create a new Vision instance

   ```
   //Pass the API key to the the Vision initializier and create the instance
   visionBuilder.setVisionRequestInitializer(new VisionRequestInitializer("API KEY"));
   Vision vision = visionBuilder.build();
   ```

3. We need to create new thread to handle networking in the background

```
//AsyncTask or java.util.concurrent (AsyncTask is deprecated in API level 30)
new AsyncTask<Object, Void, BatchAnnotateImagesResponse>() {
  protected BatchAnnotateImagesResponse doInBackground(Object... objects) {}
  protected void onPostExecute(BatchAnnotateImagesResponse
batchAnnotateImagesResponse) {}
}
```

4. Create `Feature` that we will use for Vision API.

   There are several feature types:

   - `CROP_HINTS`: Determine suggested vertices for a crop region on an image.
   - `DOCUMENT_TEXT_DETECTION`: Perform OCR on dense text images, such as documents (PDF/TIFF), and images with handwriting.
   - `FACE_DETECTION`: Detect faces within the image.
   - `IMAGE_PROPERTIES`: Compute a set of image properties, such as the image's dominant colors.
   - `LABEL_DETECTION`: Add labels based on image content.
   - `LANDMARK_DETECTION`: Detect geographic landmarks within the image.
   - `LOGO_DETECTION`: Detect company logos within the image.
   - `OBJECT_LOCALIZATION`: Detect and extract multiple objects in an image.
   - `SAFE_SEARCH_DETECTION`: Run SafeSearch to detect potentially unsafe or undesirable content.
   - `TEXT_DETECTION`: Perform Optical Character Recognition (OCR) on text within the image. Text detection is optimized for areas of sparse text within a larger image.
   - `WEB_DETECTION`: Detect topical entities such as news, events, or celebrities within the image, and find similar images on the web using the power of Google Image Search.

```
//Use a list to contain all the Features we will use
List<Feature> features = new ArrayList<>();
//Add label detection feature
Feature labelDetection = new Feature();
labelDetection.setType("LABEL_DETECTION");
labelDetection.setMaxResults(10); //Limite the respond only contains 10 results
features.add(labelDetection);
```

5. Prepare the image (Use BASE64 to encode the image)

   DO NOT USE IMAGE THAT SIZE > 10MB OR THE REQUEST WILL TIME OUT

```
//Vision API accepts BASE64 encoded image
//Covert bitmap to BASE64 encoding
private Image getBase64Image(Bitmap bitmap) {
    Image image = new Image(); //This Image type is from google api library
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
    bitmap.compress(Bitmap.CompressFormat.JPEG, 100, byteArrayOutputStream);
    byte[] bytes = byteArrayOutputStream.toByteArray();
    image.encodeContent(bytes);
    return image;
}

Image base64Image = getBase64Image(bitmap);
```

6. Prepare the Vision API request

```
//BatchAnnotateImagesRequest only accept list of AnnotateImageRequest, even there is only one image
List<AnnotateImageRequest> imageRequestList = new ArrayList<>();
AnnotateImageRequest annotateImageRequest = new AnnotateImageRequest();
annotateImageRequest.setImage(base64Image);
annotateImageRequest.setFeatures(features);
imageRequestList.add(annotateImageRequest);
//Create new batchAnnotateImagesRequest (This supports upload multiple images to Cloud)
BatchAnnotateImagesRequest batchAnnotateImagesRequest = new
BatchAnnotateImagesRequest();
batchAnnotateImagesRequest.setRequests(imageRequestList);
```

7. Call the Cloud Vision API

```
Vision.Images.Annotate request =
vision.images().annotate(batchAnnotateImagesRequest);
request.setDisableGZipContent(true);
return request.execute(); //This will return BatchAnnotateImagesResponse type
```

8. Process the response
    - Get the result for **Label**: `getLabelAnnotations()`

```
StringBuilder message = new StringBuilder();
List<EntityAnnotation> labels =
response.getResponses().get(0).getLabelAnnotations();
//Since there are different labels (Max: 10), we need for-each to get every
label
//If the labels == null, then there will be NO result returned from API
if (labels != null) {
    for (EntityAnnotation label : labels) {
        message.append(label.getDescription()).append(": "); //Get the name for
the label
        message.append(label.getScore()).append("\n"); //Get the confidence
score fot the label
    }
} else message.append("No result\n");
```

- Get the result for **Text**:

```
//To get all the text at once: getFullTextAnnotation()
TextAnnotation texts = response.getResponses().get(0).getFullTextAnnotation();
texts.getText() //Transfrom TextAnnotation to String

//To get each word in the text: getTextAnnotation()
List<TextAnnotation> texts =
response.getResponses().get(0).getTextAnnotation();
//Use for-each to iterate through the list
```

- Get the result for **Face**: `getFaceAnnotations()`

```
List<FaceAnnotation> faces =
response.getResponses().get(0).getFaceAnnotations();
//You can get the emotion status
getJoyLikelihood(); getAngerLikelihood(); getSurpriseLikelihood();
getSorrowLikelihood;
//Wearing head wear
getHeadwearLikelihood();
```

- Get the result for **Object**: `getLocalizedObjectAnnotations()`

```
List<LocalizedObjectAnnotation> objects =
response.getResponses().get(0).getLocalizedObjectAnnotations();
for (LocalizedObjectAnnotation object : objects) {
    object.getName(); //Get the name of the object
    object.getScore(); //Get the confidence score for the object
}
//Get the position of the object (You will get 4 points to draw a rectangle
around the object)
List<NormalizedVertex> v = object.getBoundingPoly().getNormalizedVertices();
v.get(index).getX(); //Get the X coordinate
v.get(index).getY(); //Get the Y coordinate
```

- Get the result for **Safe Search**: `getSafeSearchAnnotation()`

```
SafeSearchAnnotation annotation =
response.getResponses().get(0).getSafeSearchAnnotation();
//There are five categories: (adult, spoof, medical, violence, and racy)
annotation.getAdult(); annotation.getMedical(); annotation.getSpoof();
annotation.getViolence(); annotation.getRacy(); //The results are: UNLINKELY,
VERY_UNLIKELY, LIKELY etc.
```

- Get the result for **Logo**:

```
List<EntityAnnotation> entityAnnotationList =
response.getResponses().get(0).getLogoAnnotations();
entity.getDescription(); //Get the name
entity.getScore(); //Get the confidence score
```

- Get the result for **Landmark**:

```
List<EntityAnnotation> entityAnnotationList =
response.getResponses().get(0).getLandmarkAnnotations();
//Inside the for-each
entity.getDescription(); //Get the name for landmark
//If you also want the geo-locattion
LocationInfo info = entity.getLocations().listIterator().next();
info.getLatLng() //return latitude and longitude
```

- Get the result for **Image Property**:

```
DominantColorsAnnotation colors =
response.getResponses().get(0).getImagePropertiesAnnotation().getDominantColors
();
//You need a for-each to iterate through each color in DominantColorsAnnotation
//Return the fraction of each color inside the image
color.getPixelFraction();
//Return the RGB value
color.getColor().getRed();
color.getColor().getGreen();
color.getColor().getBlue();
```

- Get the result for **Web Detection**:

```
WebDetection annotation = response.getResponses().get(0).getWebDetection();
//For example: I upload a image of Poland Spring Water Bottle

//Results like: Water: 0.8541, Bottle: 0.7188, Poland Spring Brand Water:
0.7176, etc...
for (WebEntity entity : annotation.getWebEntities()) {
    entity.getDescription(); //Get the name for possible result
    entity.getScore(); //Get the confidence score
}

annotation.getBestGuessLabels(); //Will return the best match: Poland Spring
Water Bottle
```

Get the webpages that have this image:

```
//All the methods return a list of WebImage
//Use getURL() get the URLs of websites
getPagesWithMatchingImages();
getPartialMatchingImages();
getFullMatchingImages();
getVisuallySimilarImages();
```

Extra Info

1. The document for Google Cloud Vision API on Android is **lacking**. The official example was created on **2018** when the Vision API first released to public, and it cannot run on newer Android system.
    - It also pretty hard to find tutorial or example for Android on other websites.
    - But the document for **command line** is plenty.
2. The Vision API accepts up to **2000** image files. A larger batch of image files will return an error.
    - **online (synchronous)** requests - An online annotation request (`images:annotate` or `files:annotate`) immediately returns inline annotations to the user. Online annotation requests limit the amount of files you can annotate in a single request. With an `images:annotate` request

you can only specify a small number of images (<=16) to be annotated ; with a `files:annotate` request you can only specify a single file and specify a small number of pages (<=5) in that file to be annotated.

- **offline (asynchronous)** requests - An offline annotation request (`images:asyncBatchAnnotate` or `files:asyncBatchAnnotate`) starts a long-running operation (LRO) and does not immediately return a response to the caller. When the LRO completes, annotations are stored as files in a Cloud Storage bucket you specify. A `images:asyncBatchAnnotate` request allows you to specify up to 2000 images per request ; a `files:asyncBatchAnnotate` request allows you to specify larger batches of files and can specify more pages (<=2000) per file for annotation at a single time than you are able to with online requests.

3. It is very **dangerous** to hard code the API key direct into the code. So, I **restrict** the API key to only this android app: `com.example.googlecloudvisionapi` and it can only access Google Cloud Vision API.
4. Price Table:

Feature	Price per 1000 units	Price per 1000 units	Price per 1000 units
	First 1000 units/month	Units 1001 - 5,000,000 / month	Units 5,000,001 and higher / month
Label Detection	Free	$1.50	$1.00
Text Detection	Free	$1.50	$0.60
Document Text Detection	Free	$1.50	$0.60
Safe Search (explicit content) Detection	Free	Free with Label Detection, or $1.50	Free with Label Detection, or $0.60
Facial Detection	Free	$1.50	$0.60
Facial Detection - Celebrity Recognition	Free	$1.50	$0.60
Landmark Detection	Free	$1.50	$0.60
Logo Detection	Free	$1.50	$0.60
Image Properties	Free	$1.50	$0.60
Crop Hints	Free	Free with Image Properties, or $1.50	Free with Image Properties, or $0.60
Web Detection	Free	$3.50	Contact Google for more information
Object Localization	Free	$2.25	$1.50

If your application made the following requests in a particular month:

- **700** images with label detection
- **5300** images with landmark detection

Your cost would be:

- $0 for 700 label detection requests.
- $0 for the first 1000 landmark detection requests.
- $7.50 for the remaining 4300 landmark detection requests. Pricing is calculated in 1000-request blocks. For example, exactly 4000 requests is priced at 4 * $1.50. Any number of requests between 4001 and 5000 (including the 4300 requests in this example) moves the total into the next (5th) block of 1000 and is priced accordingly, adding another $1.50 to the existing cost and bringing the total cost to 5 * $1.50, or $7.50.

Total cost is $7.50.