Richtlinie zum „Software-Sprint“

OpenMLKit – Open ML Kit für Android

Schlussbericht

Zuwendungsempfänger: Katharina Rasch

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS22S21 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kurze Darstellung der Aufgabenstellung und Motivation

Motivation

Auf maschinellem Lernen bzw KI basierende Funktionen sind mittlerweile vom Smartphone nicht mehr wegzudenken. Google Lens erkennt und übersetzt automatisch Schrift in Bildern und hilft uns so, Straßenschilder in fremden Sprachen zu lesen. Beim Fotografieren mit dem Smartphone hilft die voll integrierte Gesichtserkennung der Kamera beim Fokussieren. Die in iOS integrierte Geräuscherkennung macht Menschen mit Hörschäden auf die Türklingel aufmerksam und hilft so, die Welt barrierefreier zu machen.

Allen drei Beispielen ist gemein, dass sie proprietärer Teil des Betriebssystems bzw. der vorinstallierten Apps sind. Wir wissen nicht, welche ML Modelle verwendet werden und welche Schwachstellen/Biases diese haben. Und wir können uns nicht sicher sein, ob nicht doch Bilder, Sound, Videos zur Verarbeitung in die Cloud geschickt werden.

Offene Alternativen sind gefragt -- aber leider gar nicht so einfach zu entwickeln. Insbesondere die Verwendung von ML/KI-Modellen direkt auf dem Endgerät (ohne Cloud) ist technisch immer noch sehr anspruchsvoll und erfordert tieferes Hintergrundwissen. Genau hier setzt das Projekt an.

Herangehensweise

Ziel ist, dass Android Entwickler*innen nicht zu Closed Source Bibliotheken wie Google ML Kit greifen. Dafür muss es Open Source Lösungen geben, die genau so einfach zu verwenden sind. Die entwickelte Android-Bibliothek “Open ML Kit” versucht, genau diese Lücke schließen und ermöglicht es Android-Entwickler*innen, KI-Funktionalität wie Texterkennung mit wenigen Zeilen Code in ihre Apps einzubinden.

Zum Glück musste im Projekt dafür nicht bei Null angefangen werden. Mit Tensorflow Lite und PyTorch Mobile gibt es Bibliotheken, um Machine Learning Modelle aufs Telefon zu bringen. Deren Zielgruppe sind allerdings ML Entwickler*innen und sie verlangen einiges an Fachwissen. Projektaufgabe war es, all diese Komplexität hinter sehr einfachen Schnittstellen zu verstecken ("Bild rein -> Text raus").

Was noch wichtig ist: Alles läuft direkt auf dem Smartphone, nichts wird in die Cloud gesendet. Ziel sind kleine Modelle, so dass auch ältere Geräte unterstützt werden.

Geplante Milestones

1. Genaueres Studieren / Ausprobieren des ML Kit von google
2. Evaluation von Tensorflow Lite und PyTorch Mobile anhand einfachen Use Cases (vermutlich Bildklassifizierung); Entscheidung für eines als Basis für die Bibliothek
3. Implementierung komplexerer Use Cases um das richtige Abstraktionsniveau für die Bibliothek zu finden (sowohl Bilderkennung als auch Textverarbeitung da unterschiedliche Ansprüche an Programmierschnittstelle)
    - Bild: vmtl Erkennung von Text in Bildern
    - Text: Erkennen der Textsprache, Entity Extraction?
4. Evaluierung durch Android Entwickler*in
5. Konzept um ML community bei Umsetzung weiterer Use cases einzubinden -- Stichwort "Model zoo"
6. Falls Zeit: Integrierung weiterer Use cases

Beitrag des Projektes zu den Zielen der Förderinitative „Software-Sprint"

Open ML Kit für Android ist ein Beitrag zur Civic Tech Infrastruktur. Es erlaubt Android-Entwickler*innen, Apps für Endnutzer*innen zu entwickeln, die wichtige KI-Funktionalitäten bieten, ohne dass Online-Dienste (die damit Zugriff auf persönliche Daten und Bilder bekommen) oder proprietäre Bibliotheken (bei denen man nicht genau weiß, wie die KI-Methoden eigentlich implementiert sind) genutzt werden müssen.

Ausführliche Darstellung der Ergebnisse

Mit Ende des Förderzeitraums wurden folgende Ergebnisse veröffentlicht:

Ergebnis 1: Android-Bibliothek 'openread'

Im Unterschied zu klassischer OCR (Optical Character Recognition; für die Digitalisierung von Büchern und Dokumenten, Text ist üblicherweise schwarz auf weiß), konzentriert sich 'openread' auf die sogenannte Scene Text Recognition (komplizierte Hintergründe, Mischung aus verschiedensten Fonts und Farben; nützlich zum Beispiel wenn man Telefonnummern aus Schildern kopieren oder sich Speisekarten live übersetzen lassen will, siehe Google Lens / Google Translate Apps).

Die Texterkennung in ‘openread’ läuft folgendermaßen ab: Zunächst wird das eingeschickte Bild (Bitmap) auf 720 Pixel Breite und 960 Pixel Höhe vergrößert bzw. verkleinert. Ein erstes Deep Learning Modell, das Textfindemodell, produziert anhand dieses Bildes eine Segmentierung / Heatmap, welche für jeden Pixel des Bildes besagt, ob dieser Pixel wahrscheinlich Teil eines Buchstaben ist oder nicht. Aus dieser Heatmap werden in einem nächsten Schritt Rahmen (sogenannte Bounding Boxes) um jedes der Wörter auf dem Bild berechnet. Als nächstes wird für jedes gefundene Wort ein entsprechender Bildausschnitt mit Fokus auf dieses Wort angefertigt und in einem letzten Schritt an ein zweites Deep Learning Modell, das Textlesemodell, geschickt. Dieses gibt das gefundene Wort heraus.

Alle Schritte laufen direkt auf der Telefonhardware, keine Internetverbindung ist nötig. Für die Ausführung der Modelle auf dem Telefon wird tensorflow-lite verwendet. Der erste Schritt, das Textfindemodell, wurde bereits für die Telefonhardware optimiert und braucht auf einem Mittelklasse-Smartphone von 2019 circa eine Sekunde. Das Textlesemodell hingegen wurde noch nicht optimiert und braucht nochmal circa eine Sekunde pro gefundenes Wort.

Die Bibliothek kann mit zwei Zeilen Code als Dependency in beliebige Android Apps eingebunden werden. All die oben beschriebene Komplexität wird durch die Bibliothek verborgen. Entwickler*innen brauchen nur eine Funktion anzurufen und dieser ein Bild als Bitmap zu übergeben. Die Funktion liefert eine Liste von gefundenen Wörtern (Bounding Box und das Wort an sich als String) zurück.

Ergebnis 2: Demo-App ‘openlens‘

Eine Demo-App namens ‘openlens’ zur Demonstrations- und Dokumentationszwecken. In der App können Nutzer*innen ein Bild aus ihrer Galerie auswählen. Dann startet automatisch die Texterkennung. Gefundene Wörter werden mit einem Rahmen markiert und der gefundene Text auf das Bild geschrieben.

Eine Demo-App ist im Android-Bereich eine übliche Art, eine Bibliothek zu dokumentieren. Entwickler*innen können im Sourcecode der App direkt sehen, wie man die Bibliothek einbindet und benutzt.

Ergebnis 3: Für Telefonhardware optimiertes Textfindemodel

Grund für diese Eigenentwicklung war, dass das ursprünglich eingesetzte CRAFT-Textfindemodell nur sehr eingeschränkt auf Telefonhardware funktioniert hat. Insbesondere konnte es nur auf sehr kleinen Bildern (320x240 Pixeln) ausgeführt werden und war trotzdem sehr langsam mit circa 20 Sekunden pro Bild. Schwerwiegender als die Langsamkeit war allerdings, dass bei so kleinen Bildern nur sehr großer Text überhaupt erkennbar bleibt.

Das veröffentlichte optimierte Modell ist eine Neuimplementierung von CRAFT mit einer weitaus leichtgewichtigeren Modellarchitektur. Da leider der Code zur Generierung der nötigen

Trainingsdaten und für das eigentliche Training von den CRAFT-Autor*innen nicht veröffentlicht wurde, musste dieser neu implementiert werden (und wurde natürlich auch von mir veröffentlicht).

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Die Bibliothek und Demo-App sind bereits benutzbar. Die Texterkennung ist noch etwas langsam und auch die Erkennungsqualität noch verbesserungswürdig. Meines Wissens nach ist openread/openlens allerdings die bereits am besten funktionierende Open Source Alternative für Scene Text Recognition auf Android Smartphones.

Pläne gibt es sowohl für die Weiterentwicklung der Bibliothek als auch der App:

- Das Textlesemodell ist noch nicht für Smartphonehardware optimiert und sehr langsam. Plan ist es, hier wiederum auf eine leichtgewichtige Modellarchitektur umzusteigen. Dafür wird es allerdings nötig sein, auch dieses Modell neu zu trainieren. Glücklicherweise kann hier, im Gegensatz zum Textlesemodell, schon auf einigen Code aus der Community zurückgegriffen werden.
- Das Textfindemodell und das Textlesemodell sind nicht gut aufeinander kalibriert, was die Erkennungsrate schmälert. Um die Erkennung zu verbessern, wird es nötig sein, das Textlesemodell direkt auf Ausschnitten zu trainieren, die vom Textfindemodell berechnet wurden.
- Sobald die Erkennung verbessert / beschleunigt wurde, wird die Demo-App offiziell veröffentlicht (zur Zeit muss man die App selber kompilieren und manuell installieren). Veröffentlichung ist geplant auf jeden Fall für den F-Droid Store, potentiell auch für den Play Store, falls von Google erlaubt.

Ich werde diese Weiterentwicklungen zunächst erst einmal selbstfinanziert vorantreiben, kann mir aber mittelfristig auch vorstellen, noch andere Fördermittel für die Weiterführung der Entwicklung zu beantragen.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Alle durchgeführten Arbeiten waren für die veröffentlichten Ergebnisse relevant. Es gab keine Arbeiten, die zu keiner Lösung geführt haben bzw. nicht weiterverfolgt wurden.

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Sowohl die Bibliothek als auch die Demo-App finden sich unter https://gitlab.com/krasch/openread. Der Trainingscode für das Textlesemodell ist in der Readme verlinkt.

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Es gab eine Abweichung bei der Arbeitsplanung: Statt mindestens zwei Use Cases zu implementieren, wurde nur der Use Case “Texterkennung” implementiert. Dies hatte mehrere Gründe:

- Nach Review des Google ML Kit wurde klar, dass Texterkennung eine der wichtigsten Anwendung für ML/KI auf dem Smartphone ist. Text, der gelesen, digitalisiert, übersetzt werden muss, findet sich einfach überall.
- Gleichzeitig ist Texterkennung ein relativ komplizierter Use Case und involviert in der von mir gewählten Lösung das Zusammenspiel zweier Modelle. Priorisierung von Texterkennung hieß also für mich auch: leider keine Zeit für die Implementierung anderer Use Cases.
- Auf der anderen Seite hat Texterkennung aber den riesigen Vorteil, dass Trainingsdaten frei verfügbar sind (im Gegenteil zu vielen anderen ML Anwendungen, wo man außerhalb großer Firmen kaum Chancen hat an passende Trainingsdaten zu kommmen). Insbesondere gilt für die Modelle, die ich in meiner Bibliothek einsetze, dass sie komplett auf synthetischen Daten trainiert werden können. Konkret bedeutet das: man nimmt ein Hintergrundbild ohne Text, schreibt Text drauf und trainiert das Modell dann darauf, diesen Text wiederzufinden. So lassen sich im Prinzip unendliche Mengen an Trainingsdaten generieren.

Da es zur Zeit keine konkreten Pläne zur Einbindung weiterer Use Cases gibt, wurde noch nicht mit dem Schritt “Aufbau einer Community von Mitentwickler*innen” begonnen. Aktuell ist der Plan, die selbsttrainierten Modelle bei https://huggingface.co/ zur Verfügung zu stellen, da dort bereits eine große Gemeinschaft von KI-Entwickler*innen miteinander Modelle teilt.

Abgesehen von diesen Abweichungen wurde der Arbeitsplanung und der Kostenplanung gefolgt.

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Meine Arbeit baut auf Forschungsarbeiten des ClovaAI Lab zur Texterkennung auf. Insbesondere:

- Textfindemodell CRAFT https://github.com/clovaai/CRAFT-pytorch
- Textlesemodell https://github.com/clovaai/deep-text-recognition-benchmark