Gépi tanulás alapú aláírás hitelesítés

Bitai Bence

Konzulens: Szücs Cintia Lia

Tárgy: BSc Önálló laboratórium

A cél az volt, hogy egy kiválasztott cikk eredményeit minél hűebben reprodukáljam. Az első két héten a cikk kiválasztásával, és tanulmányozásával foglalkoztam. Végül A DeepSign: Deep On-Line Signature Verification című cikket választottam.
https://ieeexplore.ieee.org/abstract/document/9335993?casa_token=_g-OaCzHb04AAAAA:Elv89piS8bVu4rd4X3_6LKc-tJC25vxAmYhHoC9KsE8ddCQeqn_0AOmPRgpD9uJl2sX9N6HZ1bc

Többek között azért esett erre a választás, mert követelmény volt, hogy a DeepSignDb nevű adatbázist használjam, és hogy online aláírás hitelesítés legyen, ezeket ez a cikk teljesíti. Emellett a cikk a megírása idejében a legjobb eredményeket produkálta bizonyos esetekben (legalábbis a cikk írói szerint). A cikk szerzői a DeepSignDb készítői, tulajdonképpen részben ennek az adatbázisnak a bemutató cikkjéről van szó. Emellett késíztettek hozzá egy új aláírás hitelesítő rendszert amit TA-RNN nek neveztek el. A konkrét felépítéséről majd később lesz szó.

Ez után a beolvasáson kezdtem el dolgozni. Ehhez először meg kellett érteni a DeepSignDb struktúráját. Mivel több kisebb adatbázis egy egyesítésével jött létre, a fájlok elnevezései és tartalmuk sem egységes, ez nagyban megnehezíti a munkámat. Az egyes fájlokban más oszlopban vannak a featureok, valamelyikben több extra feature van a másikhoz képest. Például valamelyikben volt egy olyan oszlop is, ami szimplán annyitt jelzett, hoyg hozzáért e a toll vagy nem. Finger esteben meg nem volt pressure oszlop. Sajnos maga a fájl nem tartalmazott információt arról, hogy az oszlop milyen adatot tartalmaz, ezt a fájlt tartalmazó mappa és maga a fájl nevéből kellett kinyerni. Ezeknek köszönhetően meglepően sok idő telt el ezzel, talán ez volt a második legtöbb időt igénylő folyamat az egész során.

Ha sikerült a fájl típusát és belső elrendezését kinyerni, egy fájl beolvasása csak egy egyszerű pandas readCsv függvényhívásból, majd numpy tömbbé konvertálásból állt.

Ez után bizonyos előfeldolgozási lépések következtek. Sajnos az adatbázisban lévő hibákat a szerzők nem korrigálták, tehát olyan hibák voltak benne például, hogy a timestampek hiányoztak, ugyanaz a timestamp kétszer fordult elő, vagy a fájlok elején meg a végén nulla nyomású sorok voltak. Ezeket korrigálni tudtam viszonylag gyorsan.

A cikk az aláírás featurejeire időfüggvényekként tekint, ezekből 23 van, de én csak 14 et valósítottam meg. Ezek közül volt ami egyszerűen az x y pressure koordináta volt, a többit ezekből kellett előállítani, például deriválással.

A következő lépés az aláírások aligmentje volt dynamic time warping segítségével. Ehhez fastdtw nevű könyvtárat használtam. A neve ellenére ez a folyamat aláírásonként körülbelül két másodpercet vett igénybe, ami azért probléma, mert több mint étszázezer aláírás van összesen. Később ezért is futtattam le sokkal kevesebb aláíráással.

A dynamic time warping páronként összehasonlítja az aláírásokat, és produkál egy warping path-ot, aminél a kettő közötti távolság a legkissebb. Én euklideszi távolságot használtam, de mást is lehetne,

a cikket követtem. A cikk legnagyobb újdonsága a DeepSignDb után az volt, hogy ezt a Dynamic time warping-os alignmentet elvégezte az aláírásokon azelőtt, hogy a gépi tanulási modelnek átadta volna.

Nrmalizálni is kellett az aláírásokat, ez a teljessítményt is növeli, és a tensor flow egyébként se fogad el int adatokat. Ehhez Z-normalizációt használtam, a mi az aktuális értékből kivonja az átlagot, és elosztja a szórással.

Maga a model felépítése viszonylag gyorsan ment, megtudtam találni a szükséges rétegeket. Bi-directional gru rétegeket kellett használni, és a végén egy dense feed frwardot, ammi egy 1-est vagy egy 0át produkált, ezzel eldöntve az aláírás hitellességét. Egyszer át kellett alakítani, azért mert két külön inputja egy model nek csak a functional api segítségével lehet. Az okoztott fennakadást, hogy sokszor nem tudtam hogy hogy kell rendesen paraméterezni őket. Arra is rá kellett jöjjek, hogy szükség van két input layerre, meg hogy kell egy flatten layer a feed forward layerr előtt. A legtöbb időt az egész önlab során az vette el, hogy a modelnek sehogy se volt jó az inputok formtuma. Sajnos nagyon sokszor teljesen rossz helyen kerestem a megoldást, és sok időt beleöltem abba, hogy a korábbi részeket átalaítsam, és persze az átalakítások után se működött.

A problémák fő oka az volt, hogy a tensor flow nem támogatja a változó hosszúságú inputokat(elméletileg támogatja ragged tensorral, de ezt nem tudtam működésre bírni), így végül hallgatva a konzlensem tanácsára zero maskolást alkalmaztam. A zero maskolás abból áll, hogy megkeressük al legnagyobb tömböt, és a többi nullával feltöltjük.Ez elméletileg hatamlmas veszteséget jelennt a rendzser használhatóságával kapcsolatban, és a tanítás lefuutását is nagyban hoszzabbítja.

Maga tanítás fingernél kb 2 óráig futott, ugyanennyi ideig a dtw, a beolvasás pár perc alatt meg volt. A saját laptopomon futtattam.

A futtatáshoz megpróvbáltam bme cloudos virtuális gépet használni, de ezek ha nem volt távoli asztali kapcsolat közöttük, 3-5 óra után lefagytak, és az eredménynek elvesztek. Egyébként se lett volna célra vezető, mert 4 gigabájt memóriába csak kb 7-10 ezer aláírás fér be.

A stylus esteben kb minden kétszer annyi idő volt.

Az eredmények jóval rosszabbak mint amit a cikkben láthatunk, de ez olyan szempontból érthető, hogy elég nagy egyszerűsítékkel étem többször is.

5000 comparison, 10000 aláírás alapján futtattam le. model.Predict hívás, majd, majd EER számítás.

Equal error rate (EER) az az érték, ahol a fals acceptance rate és a false rejection rate megegyezik, aláírás hitelesítésnél általában ezt szokták használni a teljesítmény értékelésére.

Finger: 7.7%

Stylus: 2.6%

Cikk finger: 1.8%

Cikk stylus: 1.5%

A továbbiakban úgy lehetne a rendszert továbbfejleszteni, hogy zero maskolás helyett valami optimálisabb technikát használjon,een kívül több aláírással kellene futtazni, és az összes time functiont kellene használni.