**Állatok felismerése és kiértékelése**

**(fejlesztői dokumentum)**

**Bevezető**

Az egyik legösztönzőbb dolog egy új téma megtanulására vagy egy új készség elsajátítására az, hogy olyan problémát old meg, amelyet a meglévő tudásoddal nehéz megoldani. Napjainkban a gépi tanulás egy trendi téma, amely az adatokban lévő minták megtalálásával és a jövő előrejelzésére vagy a múlt elemzésére való felhasználásával foglalkozik.

Az objektumok felismerése a képeken egy lenyűgöző dolog a számítógépes látásban. Lehet, hogy az ember számára triviális feladat, de egy gép számára nem.

Az elmúlt néhány évben a gépi tanulás területe, különösen a konvolúciós neurális hálózatok hatalmas előrelépést tettek az olyan nehéz problémák megoldásában, mint a vizuális felismerés és a hangfelismerés. A kutatók folyamatos fejlődést mutattak be a számítógépes látás területén azáltal, hogy munkájukat az ImageNet, a számítógépes látás tudományos mércéje alapján validálták.

A TensorFlow és az Inception-v3 modell segítségével készült ez a kis demó Java programozási nyelven, hogy felismerje a képeken lévő objektumokat, és 1000 osztályba sorolja őket, mint például oroszlán, csimpánz....stb. A felhasznált Inception-v3 modellt az ImageNet Large Visual Recognition Challenge számára képezték ki a 2012-es adatok felhasználásával.

Az előre betanított Inception V3 modell letölthető innen: <https://storage.googleapis.com/download.tensorflow.org/models/inception_dec_2015.zip>

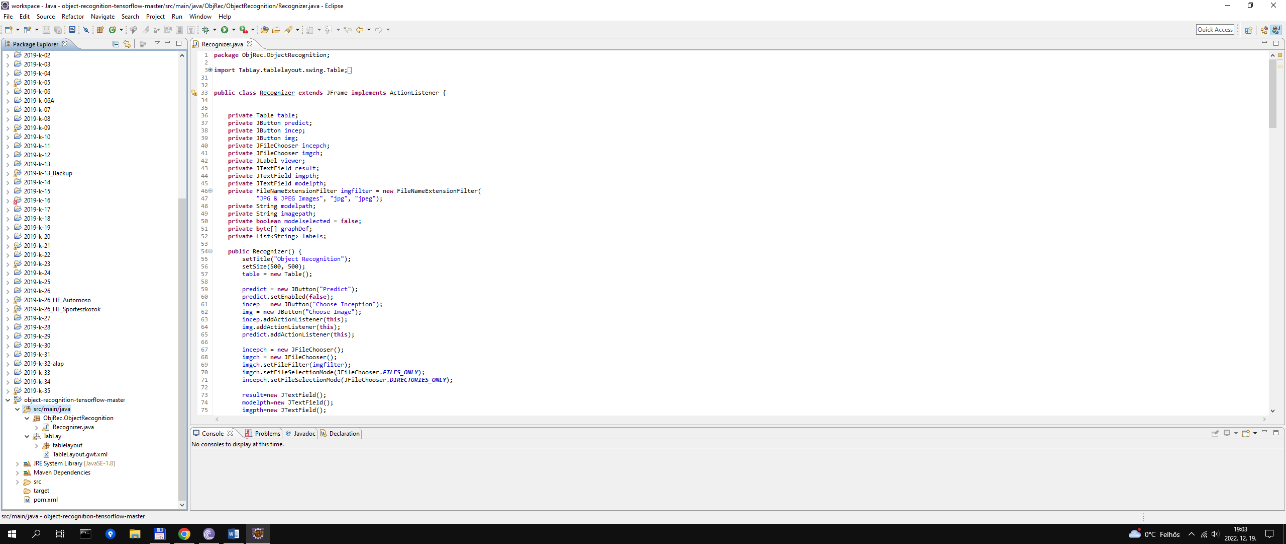
A kódot az eclipse szoftverkeretrendszerben íródott.

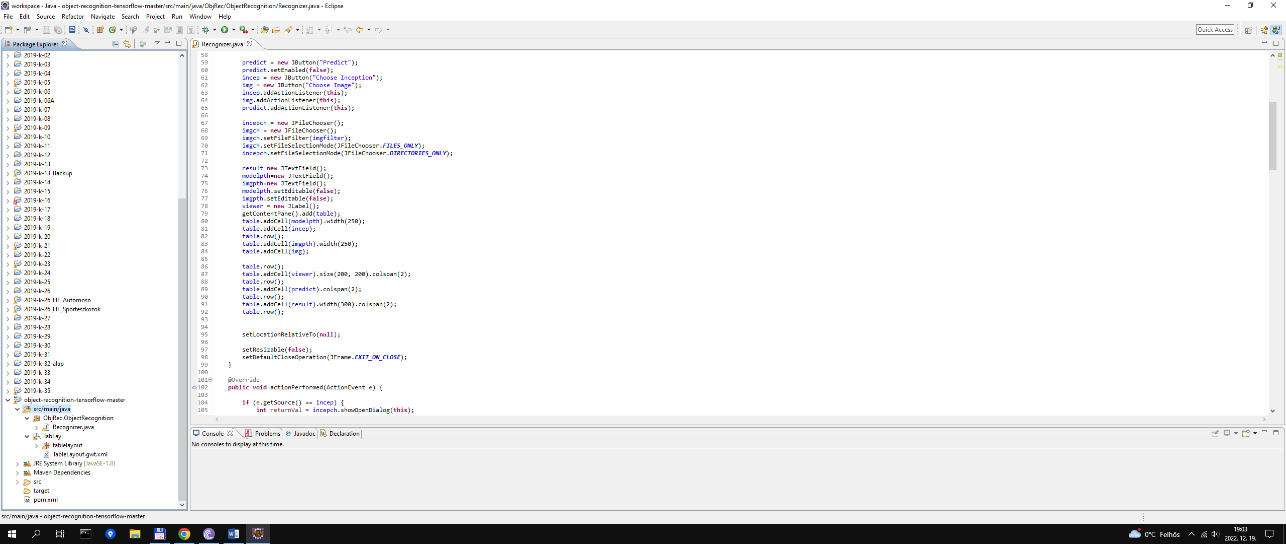
**Project**

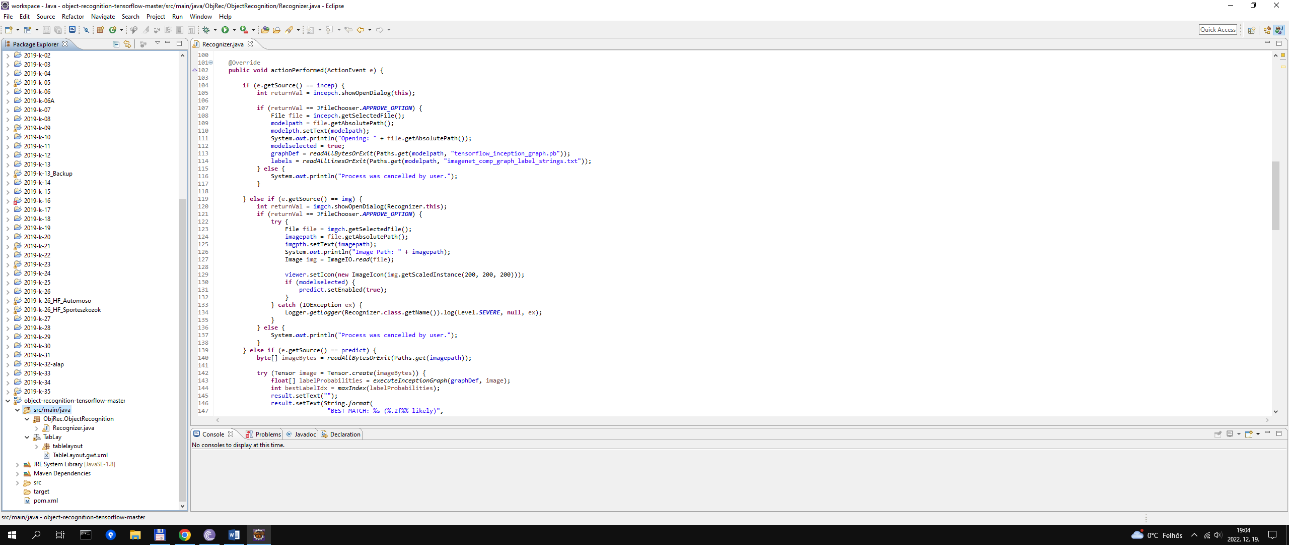
Két package-et tartalmaz a project: egy ObjRec.ObjectRecognation és egy TabLay nevezetűt.

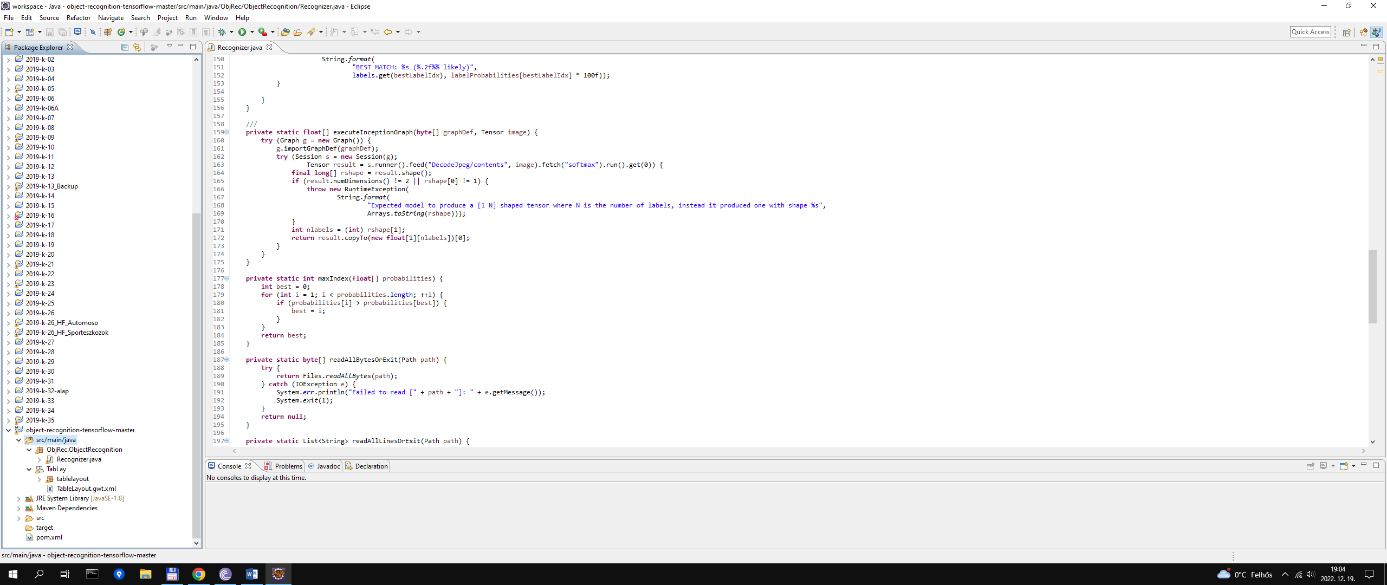
A TabLay-ben a kód lefutásakor létrejövő táblázat formai és nem-funkcionális elemeit kezelő kódsorok találhatóak.

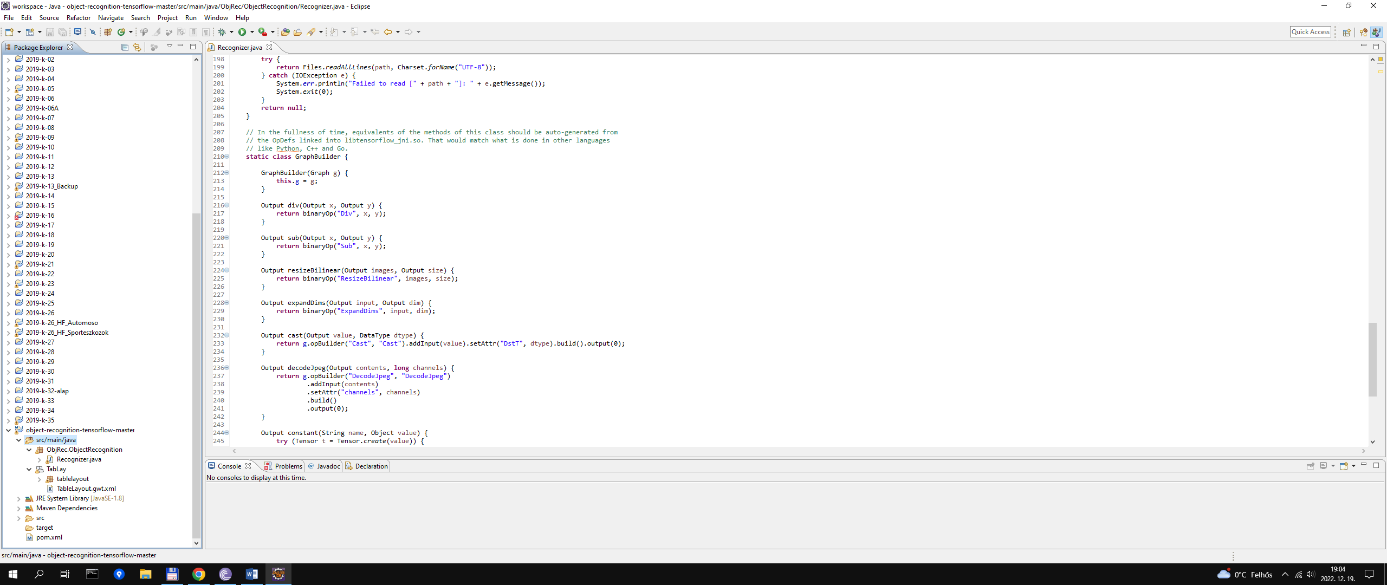
Az ObjRec.ObjectRecognation package-ben a táblázat funkcionális elemei találhatóak: a fájlok betöltésért felelősek, illetve az inception model használatáért és a várt kimenet megjelenítésért felelős metódusok.

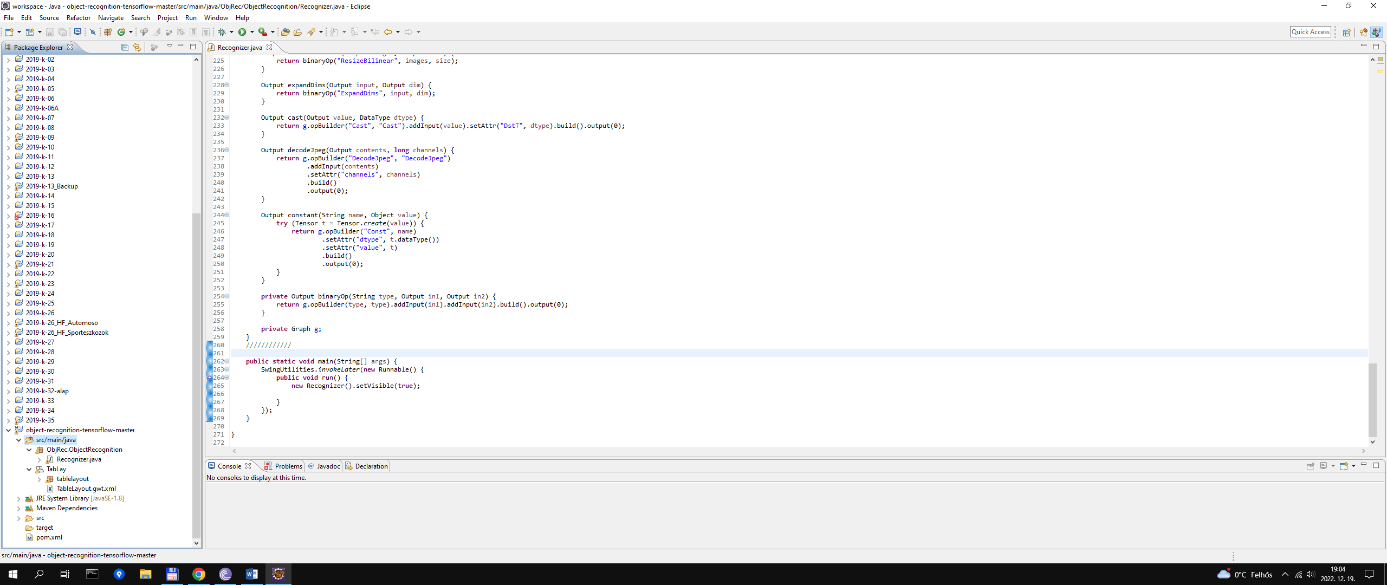












**Működés**

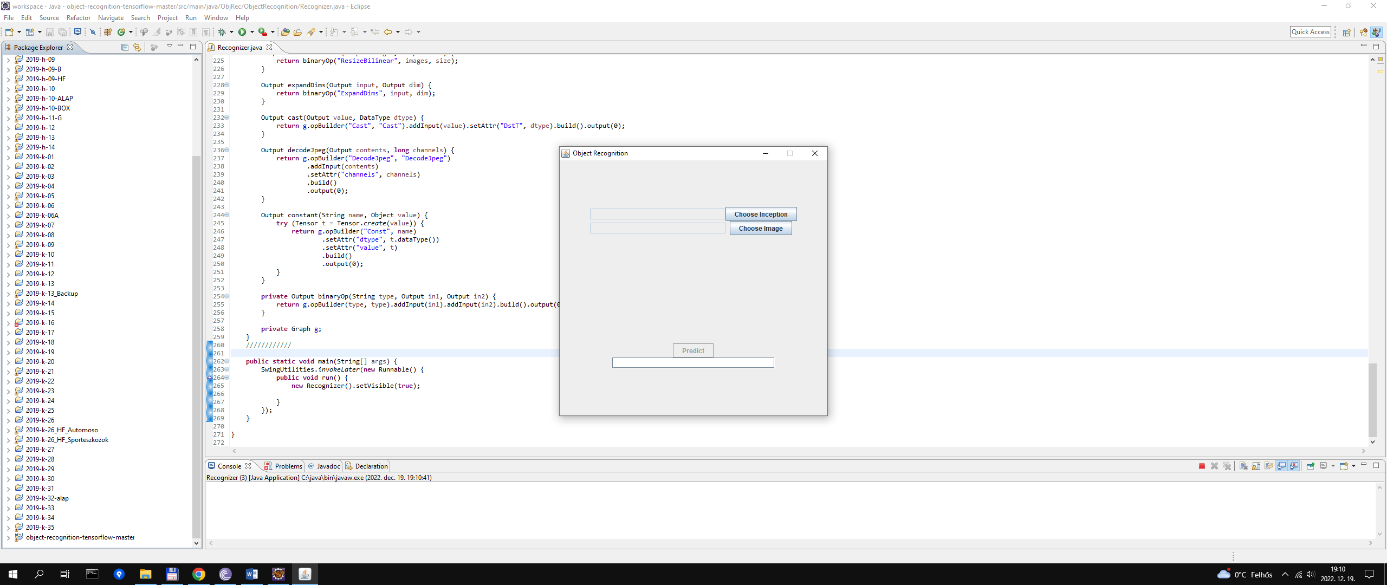
A project futtattatásához szükség van egy szoftverkeretrendszerre (eclipse, netbeans, stb…). Importálni kell a projektet. A program futásakor megjelenik egy ablak Object Recognation néven.

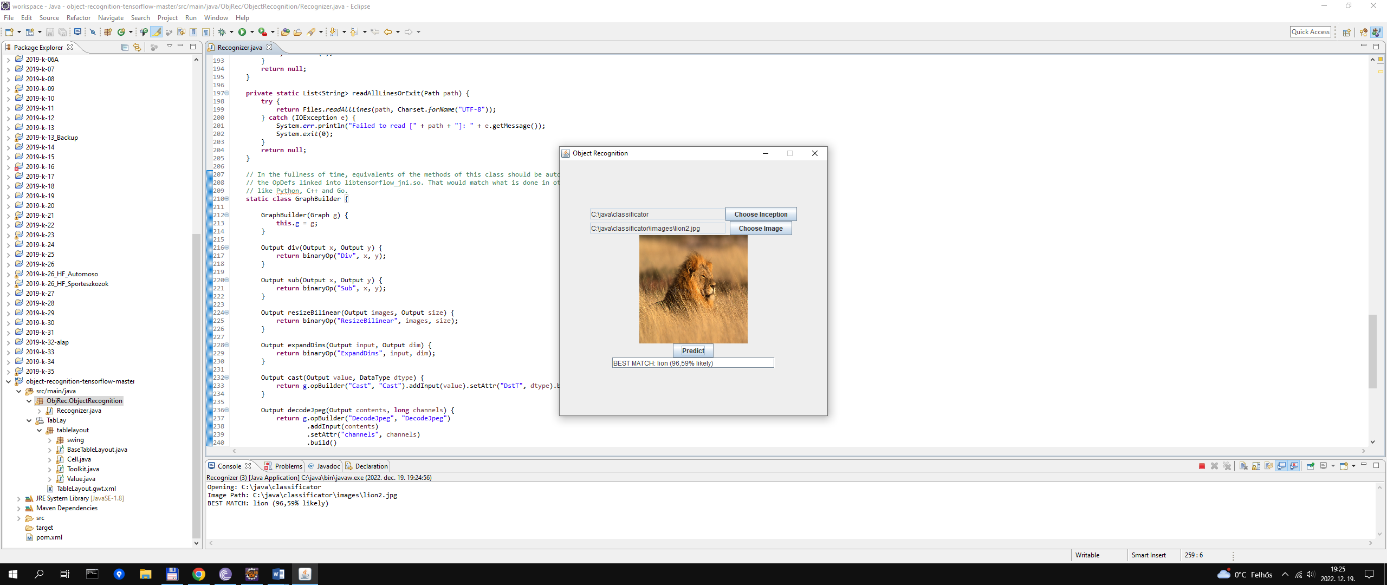
Két paramétert vár:

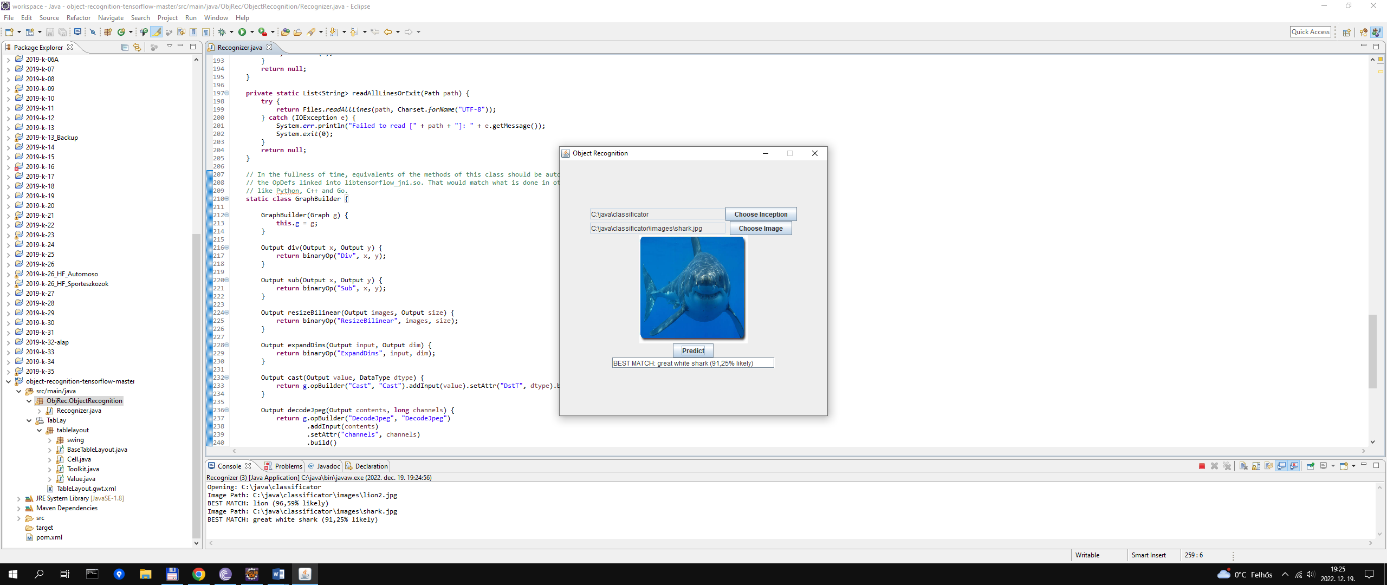
A Choose Inception az inception modell helyét várja (mivel a Recognizer fájlban a korábban említett inception V3 modell fájljait fogja használni, így érdemes a modell zip fájlját egy könnyen megtalálható helyre kibontani). Azt a mappát kell itt kiválasztani, ahove a modell zip-jét kibontotta. Hibás mappa kiválasztása esetén a program leáll.

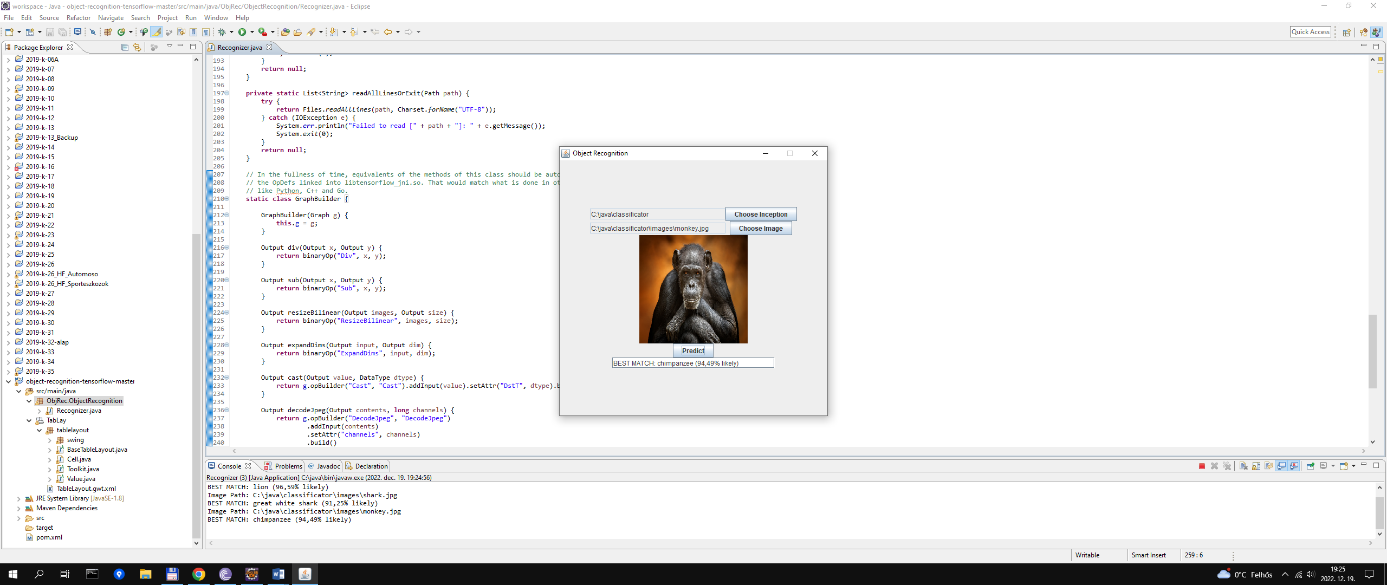
A Choose Image egy valamilyen állatot ábrázoló, png kiterjesztésű fájlt vár. Ha sikeresen kiválasztotta, meg fog jelleni a kép tartalma az ablak közepén.

A Predict a két paraméter helyes kiválasztása és a metődusok lefuttatása után ki fogja írni az állat nevét és az előre betanított és aktuális kép viszonyítási függvényében a pontosság százalékos arányát. A képek fájljait a program újraindítása nélkül cserélhetjük.









**Tensorflow**

A TensorFlow egy ingyenes és nyílt forráskódú szoftverkönyvtár a gépi tanuláshoz és a mesterséges intelligenciához. Számos feladatban használható, de különös hangsúlyt fektet a mély neurális hálózatok képzésére és következtetésére.

A TensorFlow-t a Google Brain csapata fejlesztette ki a Google belső kutatási és termelési célú felhasználására. 2015-ben az Apache License 2.0 alatt adták ki a kezdeti változatot. 2019 szeptemberében a Google kiadta a TensorFlow frissített változatát TensorFlow 2.0 néven. A TensorFlow frissített változatát TensorFlow 2.0 néven adta ki.

A TensorFlow számos programozási nyelven használható, többek között Python, JavaScript, C++ és Java nyelven. Ez a rugalmasság számos különböző ágazatban alkalmazható.

Jellemzők:

AutoDifferenciálás

Az AutoDifferenciálás egy modell gradiens vektorának automatikus kiszámítása az egyes paraméterek tekintetében. Ezzel a funkcióval a TensorFlow képes automatikusan kiszámítani a modell paramétereinek gradienseit, ami hasznos az olyan algoritmusok számára, mint például a backpropagation, amelyeknek a teljesítmény optimalizálásához gradiensekre van szükségük. Ehhez a keretrendszernek nyomon kell követnie a modell bemeneti Tenzorjaival végzett műveletek sorrendjét, majd ki kell számolnia a megfelelő paraméterek tekintetében a gradienseket. A TensorFlow-nak a modell paramétereihez tartozó grádienseket kell kiszámítania.

Buzgó végrehajtás

A TensorFlow tartalmaz egy "buzgó végrehajtási" módot, ami azt jelenti, hogy a műveletek azonnal kiértékelésre kerülnek, szemben a később végrehajtott számítási gráfhoz való hozzáadással. A buzgón végrehajtott kód lépésről lépésre vizsgálható egy hibakeresőn keresztül, mivel az adatok a kód minden egyes sorában bővülnek, nem pedig később a számítási gráfban. Ezt a végrehajtási paradigmát könnyebbnek tartják a hibakeresés szempontjából, mivel lépésről lépésre átláthatóvá teszi. A hibakeresés könnyebbé válik.

Elosztás

A TensorFlow mind az eager, mind a gráfos végrehajtásokban biztosít egy API-t a számítások több eszközre történő elosztására különböző elosztási stratégiákkal. Ez az elosztott számítás gyakran felgyorsíthatja a TensorFlow modellek képzésének és kiértékelésének végrehajtását, és ez egy gyakori gyakorlat a mesterséges intelligencia területén.

Veszteségek

A modellek képzéséhez és értékeléséhez a TensorFlow egy sor veszteségfüggvényt (más néven költségfüggvényt) biztosít. Néhány népszerű példa: átlagos négyzetes hiba (MSE) és bináris keresztentrópia (BCE). Ezek a veszteségfüggvények a modell kimenete és a várt kimenet (tágabb értelemben két tenzor közötti különbség) "hibáját" vagy "különbségét" számítják ki. A különböző adatkészletek és modellek esetében különböző veszteségeket használnak a teljesítmény bizonyos szempontjainak előtérbe helyezésére.

Mérőszámok

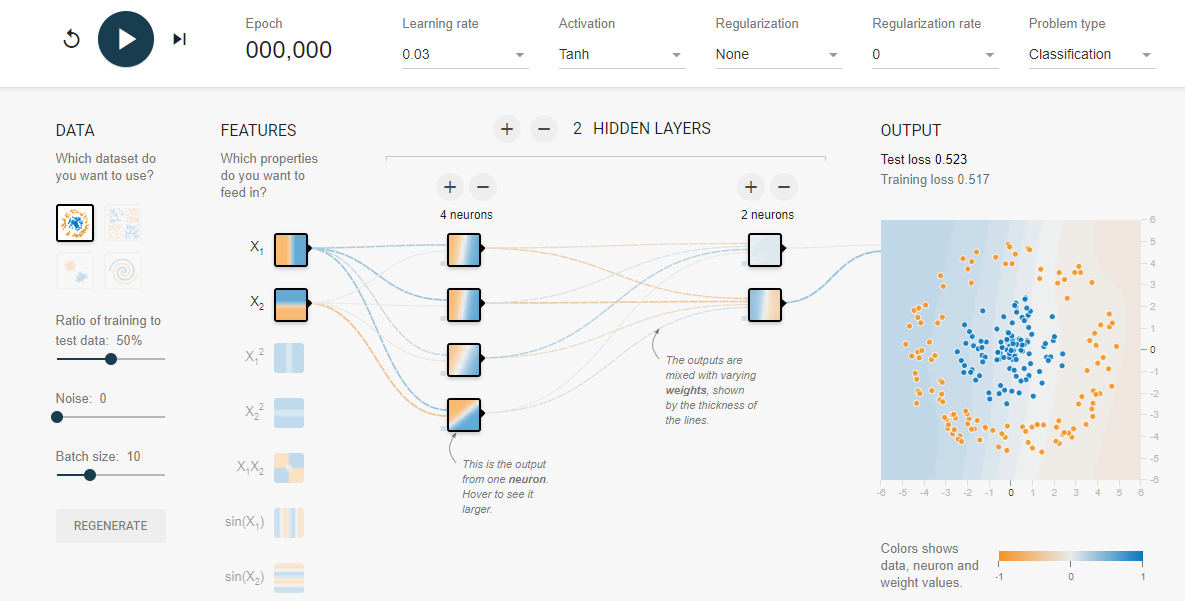
A gépi tanulási modellek teljesítményének értékeléséhez a TensorFlow API hozzáférést biztosít az általánosan használt metrikákhoz. Ilyen például a különböző pontossági metrikák (bináris, kategorikus, ritka kategorikus), valamint más metrikák, mint például a Precision, Recall és Intersection-over-Union (IoU).

TF.nn

A TensorFlow.nn egy olyan modul, amely primitív neurális hálózati műveleteket hajt végre a modelleken. Néhány ilyen művelet a konvolúciók (1/2/3D, Atrous, depthwise), aktiválási függvények (Softmax, RELU, GELU, Sigmoid, stb.) és ezek variációi, valamint egyéb Tensor műveletek (max-pooling, bias-add, stb.).

Optimalizátorok

A TensorFlow optimalizátorok sorát kínálja a neurális hálózatok képzéséhez, köztük az ADAM, ADAGRAD és a sztochasztikus gradiens leszállás (SGD). A modell képzésénél a különböző optimalizátorok különböző paraméterhangolási módokat kínálnak, amelyek gyakran befolyásolják a modell konvergenciáját és teljesítményét.



**Inception V3 model**

Az Inception v3 egy konvolúciós neurális hálózat, amely képelemzésben és objektumfelismerésben nyújt segítséget, és a Googlenet egyik moduljaként indult. Ez a Google Inception konvolúciós neurális hálózatának harmadik kiadása, amelyet eredetileg az ImageNet felismerési verseny során mutattak be. Az Inceptionv3 tervezése arra irányult, hogy mélyebb hálózatokat tegyen lehetővé, ugyanakkor a paraméterek száma ne növekedjen túlságosan nagyra: "25 millió paraméter alatt van", szemben az AlexNet 60 milliójával[1].

Ahogyan az ImageNet az osztályozott vizuális objektumok adatbázisaként is felfogható, az Inception az objektumok osztályozását segíti a számítógépes látás világában. Az Inceptionv3 architektúrát számos különböző alkalmazásban újra felhasználták, gyakran az ImageNetből "előre betanított" rendszereket használva. Az egyik ilyen felhasználási terület az élettudományok, ahol a leukémia kutatását segíti.



**Táblázat elrendezése**

A TableLayout egy egyszerű Java könyvtár a felhasználói felület widgetjeinek pozíciójának és méretének beállítására egy logikai táblázat segítségével, a HTML táblázatokhoz hasonlóan. A TableLayout magja UI toolkit-független, és támogatja a Swing, az Android és a TWL programokat. A táblázatok használatával történő elrendezés intuitív, és a TableLayout Java API-ja nagyon könnyen használható.

A TableLayout egy elágazása a libgdx-en belül található, így a TableLayout-ra nincs szükség, ha libgdx-et használunk.

Label nameLabel = new Label("Name:", skin);

TextField nameText = new TextField(skin);

Label addressLabel = new Label("Address:", skin);

TextField addressText = new TextField(skin);

Table table = new Table();

table.add(nameLabel); // Row 0, column 0.

table.add(nameText).width(100); // Row 0, column 1.

table.row(); // Move to next row.

table.add(addressLabel); // Row 1, column 0.

table.add(addressText).width(100); // Row 1, column 1.

Ez a kód 4 cellát ad a táblázathoz, amelyek két oszlopban és két sorban vannak elrendezve. Az add metódus egy Cell-t ad vissza, amely rendelkezik az elrendezés vezérlésére szolgáló metódusokkal. Itt a szövegmezők szélessége 100-ra van beállítva.

Az ebben a dokumentációban használt példakód a libgdx-re vonatkozik, de a többi támogatott eszközkészlet API-ja majdnem azonos.

Gyökér táblázat

Az UI elrendezése során egy UI widget nem állítja be a saját méretét. Ehelyett megadja a minimális, az előnyben részesített és a maximális méretet. A widget szülője a saját méretét használja ezekkel a tippekkel együtt a widget méretének meghatározásához. Sok elrendezés egyetlen táblázatot használ a gyökérben, amelynek fix mérete van, gyakran a teljes képernyő mérete. A widgetek és az egymásba ágyazott táblázatok a gyökértáblához kerülnek hozzáadásra.

A gyökértábla méretezése az egyes UI eszköztáraknál eltérő. Pl. a Swingben valószínűleg a JFrame tartalmi ablakához adnád hozzá a táblázatot. A libgdx-ben a setFillParent metódus használható:

Table table = new Table();

table.setFillParent(true);

stage.addActor(table);

Hibakeresés

A TableLayout képes hibakeresési vonalakat rajzolni, hogy láthatóvá tegye, mi történik az elrendezésben. A hibakeresést a debug meghívásával lehet engedélyezni a táblázaton. A libgdx automatikusan megjeleníti a hibakeresési vonalakat, ha azok engedélyezve vannak. Más UI eszköztárak megkövetelhetik egy metódus meghívását a vonalak megjelenítéséhez.

table.debug(); // Turn on all debug lines (table, cell, and widget).

table.debugTable(); // Turn on only table lines.

Cellák hozzáadása

A widgetek hozzáadása egy táblázathoz az add metódussal történik (az olyan UI eszköztárak esetében, amelyek már rendelkeznek add metódussal, az addCell metódust használjuk). Ez egy cellát ad hozzá az aktuális sorhoz. A következő sorra való áttéréshez hívjuk a row metódust.

table.add(nameLabel); // Row 0, column 0.

table.add(nameText); // Row 0, column 1.

table.row(); // Move to next row.

table.add(addressLabel); // Row 1, column 0.

table.add(addressText); // Row 1, column 1.

A hozzáadási módszer egy Cell-et ad vissza, amely rendelkezik az elrendezést szabályozó tulajdonságokkal. A cellára vonatkozó minden metódus visszaadja a cellát, lehetővé téve a hívások láncolását.

table.add(nameText).padLeft(10).width(100); // Sets left padding and width on the new cell.

Logikai táblázat

A cellák egy logikai táblázatot alkotnak, de ez nem a táblázat widgethez van méretezve.

A külső kék téglalap a táblázat widget méretét mutatja. A belső kék téglalap a logikai táblázat méretét mutatja, amely alapértelmezés szerint középre van igazítva. Az igazítás a táblázaton lévő metódusokkal módosítható. A táblázat metódusok visszaadják a táblázatot, így a cellamódszerekhez hasonlóan láncolhatók.

table.right().bottom();

A cellák tulajdonságai

Expand

Ahhoz, hogy a logikai táblázat a táblázat widget teljes méretét elfoglalja, a TableLayoutnak meg kell mondani, hogy mely cellák kapják meg az extra helyet.

table.add(nameLabel).expandX(); // Column 0 receives all extra horizontal space.

table.add(nameText).width(100);

table.row();

table.add(addressLabel);

table.add(addressText).width(100);

A piros vonalak a cellák határait, a zöld vonalak pedig a widgetek határait mutatják. Vegyük észre, hogy a bal oldali oszlop az x irányban az összes extra helyet megkapta. Csak egy cellának kell kitágulnia ahhoz, hogy az egész oszlop vagy sor kitáguljon. Ha több oszlop bővül, az extra hely egyenletesen oszlik el.

table.add(nameLabel).expandX(); // Receives extra horizontal space.

table.add(nameText).width(100).expandX(); // Also receives extra horizontal space.

table.row();

table.add(addressLabel);

table.add(addressText).width(100);

Expand also works in the y direction via the expandY method. The expand method causes expand to happen in both directions.

table.add(nameLabel).expand(); // Receives all extra horizontal and vertical space.

table.add(nameText).width(100);

table.row();

table.add(addressLabel);

table.add(addressText).width(100);

Kiegyenlítés

A logikai táblázat igazításához hasonlóan egy widget is igazítható a cellán belül.

table.add(nameLabel).expand().bottom().right(); // Aligned bottom right.

table.add(nameText).width(100).top(); // Aligned top.

table.row();

table.add(addressLabel);

table.add(addressText).width(100);

Kitöltés

A kitöltési módszer hatására egy widget a cellához méretezetté válik. Az expand-hoz hasonlóan a fillX és fillY metódusok is léteznek.

table.add(nameLabel).expand().bottom().fillX(); // Sized to cell horizontally.

table.add(nameText).width(100).top();

table.row();

table.add(addressLabel);

table.add(addressText).width(100);

Widget mérete

Alapértelmezés szerint a táblázat megpróbálja a widgeteket a kívánt méretre méretezni. Ha a widgetek nem férnek el, akkor a kívánt méret és a minimális méret között méretezzük őket, a nagyobb kívánt mérettel rendelkező widgetek pedig több helyet kapnak. Ha a widgetek nem férnek el a minimális méretükön, akkor az elrendezés megszakad, és a widgetek átfedhetik egymást. A kitöltési módszerek nem teszik nagyobbá a widgetet a widget maximális méreténél.

A widgeteket nem szabad alosztályozni a preferált, a minimális vagy a maximális méret megváltoztatására. Ehelyett ezeket a méreteket a cellán lehet beállítani, és a widget értéke helyett ezeket fogja használni.

table.add(nameLabel);

table.add(nameText).minWidth(100); // Sets min width.

table.row();

table.add(addressLabel);

table.add(addressText).prefWidth(999); // Sets pref width.

Itt a 999 prefWidth értéke nagyobb, mint a táblázaté, ezért a táblázat méretét lecsökkentjük.

A width egy rövidített módszer a minWidth, prefWidth és maxWidth azonos értékre történő beállítására. A height egy rövidített módszer a minHeight, prefHeight és maxHeight azonos értékre történő beállítására. A size metódus egy szélességet és egy magasságot vesz fel, és mind a hat tulajdonságot beállítja.

Padding

A padding a cella szélei körüli extra hely.

table.add(nameLabel);

table.add(nameText).width(100).padBottom(10); // Sets bottom padding.

table.row();

table.add(addressLabel);

table.add(addressLabel);

table.add(addressText).width(100).pad(10); // Sets top, left, bottom, right

Vegye figyelembe, hogy a cellák közötti kitöltés kombinálódik, így a szövegmezők között 20 pixel van. A hibakeresési sorok nem feltétlenül mutatják, hogy a kitöltés melyik cellából származik, mivel ez nem fontos a táblázat elrendezése szempontjából.

A táblázat szélei is kitölthetők.

table.pad(10);

Távolságtartás

A kitöltéshez hasonlóan a szóköz a cella szélei körüli extra tér. A cellák közötti távolság azonban nem kombinálódik, hanem a kettő közül a nagyobbat használja. Továbbá a távolságtartás nem a táblázat szélénél kerül alkalmazásra. A távolságtartás megkönnyíti, hogy a cellák között egységes térköz legyen.

table.add(nameLabel);

table.add(nameText).width(100).spaceBottom(10); // Sets bottom spacing.

table.row();

table.add(addressLabel);

table.add(addressText).width(100).space(10); // Sets top, left, bottom, right spacing.

Vegye figyelembe, hogy a cellák közötti távolság nem kombinálódik, így a szövegmezők között 10 pixel van. Azt is vegye figyelembe, hogy az alsó szövegmező alatt nincs távolságtartás, mivel a táblázat szélén nem alkalmazunk távolságtartást.

Colspan

Egy cella több oszlopot is felölelhet.

table.add(nameLabel);

table.add(nameText).width(100).spaceBottom(10);

table.row();

table.add(addressLabel).colspan(2); // Spans 2 columns.

Vegye figyelembe, hogy nincs sortávolság. Ennek eléréséhez használjon beágyazott táblázatot.

Uniform

A true értékre állított uniform cellák azonos méretűek lesznek.

table.add(nameLabel).uniform(); // These two cells will have

table.add(nameText).width(100).uniform(); // the same width and height.

table.row();

table.add(addressLabel);

table.add(addressText).width(100);

Alapértelmezettek

Cell alapértelmezett értékek

Gyakran sok cellának ugyanazok a tulajdonságai, ezért az összes cella alapértelmezett tulajdonságainak beállítása jelentősen csökkentheti az elrendezéshez szükséges kódot. A táblázat defaults metódusa egy olyan cellát ad vissza, amelynek tulajdonságai az összes cella alapértelmezett tulajdonságai.

table.defaults().width(100); // Sets defaults for all cells.

table.add(nameLabel);

table.add(nameText);

table.row();

table.add(addressLabel);

table.add(addressText);

Oszlop alapértelmezett értékek

A táblázat columnDefaults metódusa egy olyan cellát ad vissza, amelynek tulajdonságai az adott oszlop összes cellájának alapértelmezett tulajdonságai. Az itt beállított tulajdonságok felülírják a cellák alapértelmezett tulajdonságait. Az oszlopok indexelése 0-tól kezdődik.

table.columnDefaults(1).width(150); // Sets defaults for cells in column 0.

table.add(nameLabel);

table.add(nameText);

table.row();

table.add(addressLabel);

table.add(addressText);

Alapértelmezett sorok

A sor metódus meghívásakor egy olyan cellát ad vissza, amelynek tulajdonságai az adott sor összes cellájának alapértelmezett tulajdonságai. Az itt beállított tulajdonságok felülírják mind a cella alapértelmezett tulajdonságait, mind az oszlopok alapértelmezett tulajdonságait. Megjegyzendő, hogy a row meghívása megengedett a cellák hozzáadása előtt. Ez lehetővé teszi, hogy az első sornak sor alapértelmezett tulajdonságai legyenek.

table.row().height(50); // Set cell defaults for row 0.

table.add(nameLabel);

table.add(nameText);

table.row().height(100); // Set cell defaults for row 1.

table.add(addressLabel);

table.add(addressText);

Halmok

A verem widget egy speciális tároló, amely minden egyes gyermeket a verem méretének megfelelő méretűre rendez. Ez akkor hasznos, ha egymásra rakott widgetekre van szükség. A veremhez elsőként hozzáadott widget alulra, az utoljára hozzáadott widget pedig felülre rajzolódik.