**Dokumentáció**

**1. Bevezetés**

Egy olyan programot akartunk készíteni Python programozási nyelven, ami képes egy előre betanított modell használatával bináris osztályozási feladatot old meg. Képek alapján eldönti, hogy azon egy kutya vagy macska szerepel.

A rendszer a Kaggle-ről letöltött előre betanított modellt használja, amely képes a bemeneti képeket feldolgozni és predikciókat adni.

A dokumentáció következő részében a kód működése található lépésről lépésre, megvizsgálva az esetleges hibalehetőségeket, végül javaslatokat írunk a további fejlesztésekre.

**2. A kód működése**

**2.1. Modell letöltése és betöltése**

A kód a kagglehub modul segítségével tölti le a „cat-vs-dog” modellt:

1. path = kagglehub.model\_download("bruce0000/cat-vs-dog/keras/default")

Ez az utasítás a Kaggle API-ját használja, amely a modellfájlokat az alapértelmezett gyorsítótár könyvtárba helyezi.

Az előre betanított modellt a keras könyvtár load\_model függvényével tölti be:

1. loaded\_model = load\_model("/root/.cache/kagglehub/models/bruce0000/cat-vs-dog/keras/default/1/cat\_vs\_dog\_new.keras")

Ez az utasítás biztosítja, hogy a modell használatra kész legyen, és alkalmas legyen képek kategorizálására.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírásSzükséges fájlkönyvtár elrendezése:

Vannak bonyolultabb modellek, ahol az előfeldolgozáshoz szükséges külön mappákba rendezni külön a tanító halmazt, és külön a tesztelő (validációs) halmazt.

A mi esetünkben elég megadni a fájl elérési útvonalát, hiszen a betanításra nincs szükség.

**2.2. Képek előfeldolgozása**

A képek előfeldolgozása kulcsfontosságú, mivel a modell egy adott bemeneti formátumot vár el.

A preprocess\_image függvény a következő lépéseket hajtja végre:

1. **Kép beolvasása**: A PIL (Python Imaging Library) nyitja meg a képet.
2. **Szürkeárnyalatossá alakítás**: A kép konvertálása szürkeárnyalatossá történik a memóriahasználat csökkentése érdekében.
3. **Képméret módosítása**: A bemeneti képet 128×128 pixeles méretre alakítja át, hogy kompatibilis legyen a modell elvárásaival.
4. **Normalizálás**: A képpontértékek [0, 1] közé skálázódnak a következő módon:

1. img\_array = np.array(img) / 255.0

Az előfeldolgozott képből a modell által elvárt 4 dimenziós mátrix készül (batch méret, szélesség, magasság, csatornák száma):

1. preprocessed\_image = np.expand\_dims(preprocessed\_image, axis=0)

**2.3. Predikció**

A modell a kategorizálást a predikciós függvénnyel végzi:

1. predictions = loaded\_model.predict(preprocessed\_image)

A kimenet egy valószínűségi érték (0 és 1 között). A kategória meghatározása egy 0.5-ös küszöbértékkel történik:

* **Érték < 0.5**: Kutya
* **Érték ≥ 0.5**: Macska

A végső kategória megjelenítéséhez a következő logika érvényesül:

1. predicted\_label = int(predictions[0][0] < 0.5)

2. pred\_label\_text = 'dog' if predicted\_label == 0 else 'cat’

**2.4. Eredmények vizualizációja**

A predikció eredményének szemléltetése a matplotlib segítségével történik. A bemeneti képet és a predikciót egy diagramon jeleníti meg:

1. plt.imshow(preprocessed\_image[0].reshape(128, 128), cmap='gray')

2. plt.title(f'Predicted: {pred\_label\_text}')

3. plt.axis('off')

4. plt.show()

**A képen emlős, szöveg, kutya, képernyőkép látható

Automatikusan generált leírás**

A képen emlős, szöveg, macska, Macskafélék látható

Automatikusan generált leírás

**3. Hibalehetőségek és azok kezelése**

**3.1. Hibás elérési út**

**Probléma**: Ha a megadott fájlút nem létezik, a kód nem tudja beolvasni a képet.

**Megoldás**: Ellenőrizni kell a fájlútvonal létezését:

1. if not os.path.isfile(data):

2. print("File does not exist, please check the path.")

**3.2. Nem megfelelő képméret**

**Probléma**: Ha a kép mérete nem kompatibilis a modellel, a kód futás közben hibát jelezhet.

**Megoldás**: Az előfeldolgozás során automatikusan méretezzük át a képet.

**3.3. Alacsony predikciós pontosság**

**Probléma**: A modell nem nyújt megfelelő eredményeket, ha az adatbázis nem reprezentatív.

**Megoldás**: A modell további finomhangolása és több adatminta használata.

**3.4. Szürkeárnyalatos képek korlátai**

**Probléma**: Az RGB képek több információt hordozhatnak, mint a szürkeárnyalatosak.

**Megoldás**: Az előfeldolgozás módosítása az RGB csatornák kezelésére.

**4. Továbbfejlesztési lehetőségek**

1. **Színes képek kezelése**:

A szürkeárnyalatos helyett RGB képek használatával növelhető a pontosság.

1. **Adatbővítés (Data Augmentation)**:

Olyan technikák bevezetése, mint forgatás, tükrözés, eltolás vagy zaj hozzáadása, növeli a modell általánosíthatóságát.

1. **Valós idejű predikció**:

Webkamerából származó valós idejű képek feldolgozása és azonnali kategorizálás.

1. **Több kategória felismerése**:

A modellt tovább lehet képezni több kategóriára, például különböző kutyafajták vagy egyéb állatok felismerésére.

1. **Mobil eszközök támogatása**:

A modell optimalizálása mobilalkalmazásokhoz TensorFlow Lite vagy ONNX formátumban.

**5. Következtetés**

Ez a kód egyszerű, de hatékony módon mutatja be, hogyan lehet előre betanított modellt alkalmazni képfelismerési feladatokhoz.

A rendszer a beolvasástól a predikcióig minden lépést lefed, és könnyen bővíthető. A továbbfejlesztési javaslatok segíthetnek abban, hogy az alkalmazás pontosabb és szélesebb körben használható legyen.