Képek betöltése és előfeldolgozása

Képalkotási hibák detektálása

A program a képeken gyakran előforduló képalkotási hibákat próbálja meg felismerni. A detektálás eredményéről a felhasználó értesítést kap a grafikus felületen található szövegdobozon keresztül. A képalkotási hibák detektálása függvények segítségével történik, amelyek a fault\_detection.py fájlban találhatóak meg.

Expozíció

A képek expozíciós állapota az estimate\_exposure függvény segítségével történik. A képek feldolgozása során gyakran előfordul, hogy a felvétel túl sötét (alul exponált) vagy túl világos (túlexponált), ami rontja a vizuális minőséget és az utófeldolgozás hatékonyságát. Ez a függvény a kép fényességi eloszlását elemzi, különböző szempontok alapján, hogy objektív módon meghatározza, melyik kategóriába sorolható a kép.

A függvény bemeneti paramétere egy színes kép numpy tömb formátumban (BGR színtérrel rendelkezik, amely az OpenCV által támogatott formátum).

A függvény három fő tényezőt vizsgál a kép expozíciójának meghatározásához:

1. Hisztogram elemzés és a domináns fényességérték meghatározása – A szürkeárnyalatos kép fényességi eloszlásából a leggyakoribb fényességi értéket (módusz) keresi meg.
2. Átlagos fényesség kiszámítása – A teljes kép fényességeloszlását figyelembe véve meghatározza az átlagos fényességi értéket.
3. Sötét és világos pixelek arányának vizsgálata – Az alacsony (<50) és magas (>205) fényességi értékek arányát méri a teljes képen belül.

Ezek alapján a függvény osztályozza a képet az alábbi kategóriák egyikébe:

* Alul exponált (1): Ha a kép leggyakoribb fényességi értéke alacsony (≤120), az átlagos fényessége szintén alacsony (<100), és a sötét pixelek aránya nagyobb mint 40%, akkor a kép alul exponáltnak tekinthető.
* Megfelelő expozíció (0): Ha a leggyakoribb fényességi érték 120 és 160 között van, az átlagos fényesség 100 és 180 között mozog, és nincsen kiugróan nagy arányban sötét vagy világos pixel, akkor a kép jól exponált.
* Túlexponált (2): Ha a leggyakoribb fényességi érték magas (≥160), az átlagos fényesség is nagy (>180), és a világos pixelek aránya meghaladja a 40%-ot, akkor a kép túlexponáltnak tekinthető.

A függvény eredményeként egy logikai értéket (True vagy False) és egy egész számot ad vissza, amely az expoziciós osztályozást jelöli.

def estimate\_exposure(image: np.array):

    gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

    hist\_gray = cv2.calcHist([gray], [0], None, [256], [0, 256])

    max\_pixel\_brightness\_place = int(np.argmax(hist\_gray))

    avg\_brightness = np.mean(gray)

    dark\_pixels = np.sum(gray < 50) / gray.size

    bright\_pixels = np.sum(gray > 205) / gray.size

    if max\_pixel\_brightness\_place <= 120 and avg\_brightness <100

    and dark\_pixels > 0.4:

        return True, 1

    elif max\_pixel\_brightness\_place > 120 and

    max\_pixel\_brightness\_place < 160 and 100 <= avg\_brightness <=

    180:

        return False, 0

    elif max\_pixel\_brightness\_place >= 160 and avg\_brightness >

    180 and bright\_pixels > 0.4:

        return True, 2

    else:

        return False, 0

1. kódrészlet: estimate\_exposure függvény

Elmosódás

A képek elmosódottságának megállapítása estimate\_blur függvény segítségével történik. A függyvény feladata annak eldöntése, hogy a bemeneti kép homályos vagy éles. Amennyiben a kép homályos, a függvény annak típusát is meghatározza egy mélytanuláson alapuló osztályozó modell segítségével.

Ennek a kétlépcsős rendszer ötvözi a hagyományos képfeldolgozási technikákat a gépi tanulással, és pontosan képes felismerni az életlenség meglétét és típusát.

Működés

1. Laplace-variancia alapú homályosságvizsgálat

Az első lépésben a rendszer a kép élességét a Laplace-operátor varianciájával méri.

A számítás előtt a kép szürkeárnyalatosra konvertálása után újra méretezés történik a megadott kontans érték RESIZE\_HEIGHT pixel magasságra, miközben az oldalarányt megtartja. Ez a lépés azért fontos, mert:

* Különböző felbontású képeken a Laplace-variancia nem összehasonlítható.
* A méretezés egységesíti a feldolgozási feltételeket és javítja a küszöbérték alapú döntés pontosságát.

A Laplace-variancia értelmezése a következők szerint alakul:

* Ha a variancia magas, akkor éles a kép
* Ha a variancia alacsony, azaz kisebb a megadott küszöbértéknél, akkor a kép potenciálisan homályos, és a második lépés következik.

1. Homály típusának meghatározása neurális hálóval

Amennyiben a Laplace-variancia alapján a kép homályosnak minősül, egy előre betanított Keras-alapú konvolúciós neurális hálózat segítségével történik a homály típusának meghatározása. A hálózat a következő osztályokba sorolja a képet:

* Defókuszált – amikor a kép fókuszon kívül van
* Mozgási – amikor a kép mozgás hatására lett elmosódott

A neurális háló előfeldolgozási lépésként 512x512 pixeles képeket vár, normálva a [0, 1] tartományba

A homályosság osztályozó modell tanítása a Kaggle Blur Dataset adatkészlettel lett tanítva, amely két mappát tartalmaz: motion és defocus.

def classify\_blur\_types(img):

    resized = np.array([cv2.resize((img \* 255).astype(np.uint8), (512, 512)) / 255.0])

    preds = blur\_classifier.predict(resized, batch\_size=BATCH\_SIZE, verbose=0)

    return CLASS\_NAMES[np.argmax(preds[0])]

def resize\_keep\_aspect(image, target\_height):

    h, w = image.shape[:2]

    scale = target\_height / h

    new\_width = int(w \* scale)

    return cv2.resize(image, (new\_width, target\_height))

def laplacian\_variance(image):

    gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

    resized = resize\_keep\_aspect(gray, target\_height = RESIZE\_HEIGHT)

    lap\_var = cv2.Laplacian(resized, cv2.CV\_64F).var()

    return lap\_var

def estimate\_blur(image: np.array):

    lap\_var = laplacian\_variance(image)

    if lap\_var < LAPLACE\_THRESHOLD:

        detected\_type = classify\_blur\_types(image)

        return True, lap\_var, detected\_type

    else:

        return False, lap\_var, ""

Piros szem

Az piros szem hibával rendelkező képek detektálása az estimate\_redeye függvény segítségével történik. A függvény feladata a vörös szem effektus automatikus detektálása a bementi képen. A vörös szem hiba leggyakrabban vaku használata során jelentkezik, amikor a fény visszatükröződik a retináról. A függvény a következő lépések segítségével azonosítja az ilyen területeket:

1. Arcok és szemek detektálása a face\_detection függvény és a Haar-kaszkád osztályozó segítségével.
2. Piros szem területek azonosítása a HSV színtérben adott színküszöbök alkalmazásával.
3. Maszk generálása és vizuális ellenőrzés a piros szem területekről.
4. Pixelalapú elemzés és küszöbérték számítás a végső döntés meghozatalához a detektálás eredményéről.

A program először egy alfüggvény segítségével arcfelismerést végez a bemeneti képen, amely a YOLOv8 modell segítségével azonosítja a képen lévő arcokat. Ha nem található arc a piros szem elemzést nem hajtja végre, ami pontosabbá teszi a detektálást.

Ha arcokat talál, a cv2CascadeClassifier("haarcascade\_eye.xml") modellt használja a szemek felismerésére. Az eyeRects tömb minden szem téglalap koordinátáit és magasságát és szélességét tartalmazza, amely minden eleme [x, y, w, h] alakú. Ha nem található szem, a függvény False-t ad vissza.

A detektált szemterületeket kivágja az eredeti képből, majd HSV színtérbe konvertálja. Egy előre meghatározott színtartományt használ, hogy létrehozza a piros szem maszkot.

A függvény a maszk alkalmazása után megszámolja a piros pixeleket. Ha ezek száma meghalad egy küszöbértéket (a szem összes pixelének 1%-át), akkor piros szem hibát jelez.

Ha legalább egy szem tartalmaz vörös szem hibát, a függvény igaz értékkel tér vissza. Ellenkező esetben hamis eredményt ad vissza.

Purple Fringe:

A képeken előforduló lila rojtosodás hibák detektálása purple\_fringe\_detection függvény segítségével történik. A függvény szín- és élalapú technikák kombinációjával végzi a detektálást és opcionálisan visszaadja a hibás területeket tartalmazó bináris maszkot.

Működése:

1. Színcsatornák szétválasztása és színkülönbség kiszámítása

A függvény első lépésként szétválasztja a BGR színcsatornákat. Ezután kiszámítja a vörös és kék csatornák különbségének abszolút értékét (diff\_rb). A nagy eltérések jellemzően purple fringing jelenlétére utalnak, mivel ez a jelenség főként a piros és kék csatornákban jelenik meg.

1. Magas kontrasztú területek detektálása

A zöld és piros csatornák különbsége alapján detektálja a nagy kontrasztú részeket (high\_contrast\_area). Erre azért van szükség, mert a purple fringing jellemzően ezek mentén fordul elő.

1. Purple fringing kezdeti maszk létrehozása

A fent kiszámított színkülönbség és kontraszt alapján létrehoz egy bináris maszkot (purple\_fringing), amely azokat a pixeleket tartalmazza, amelyek mind a színkülönbség, mind a kontraszt szempontjából gyanúsak.

Ez a maszk fekete-fehér képként kerül előállításra (purple\_fringing\_image), ahol a hibás területek fehér színnel jelennek meg.

1. HSV színtér használata a színalapú szűréshez

A bemeneti képet HSV színtérbe konvertálja, amely jobban elkülöníti a színeket és megkönnyíti a lila árnyalatok célzott detektálását.

Három különböző lila árnyalatra definiál színtartományokat, ahol a Hue (H) értékek különböznek a Saturation (S) és a Value (V) egyaránt magas:

* Purple (Hue érték: 260 – 320)
* Blue-purple (Hue érték: 240 – 260)
* Red-purple (Hue érték: 320 – 360)

Ezekből egy színmaszk (color\_mask) készül, amely egyesíti a különböző tartományokat.

1. Szín és kontraszt együttes figyelembevétele

A purple\_fringing\_image és a color\_mask kombinálásával megszűri azokat a pixeleket, amelyek egyszerre szín és kontraszt alapján is hibásnak tekinthetők (combined\_mask).

Ha a kombinált maszkban található aktív pixel, akkor:

1. Kontúr- és területvizsgálat

A függvény kontúrokat keres a maszkon. Ezután kiszámítja az egyes területek méretét. Ha egy kontúr túl nagy, akkor azt nem tekinti hibának. A kontúrok területének küszöbértékét area\_threshold változó tartalmazza, amelynek értéke arányosan állítható a kép méretéhez, így skálázható különböző felbontású képekre is. Az elfogadott méretű kontúrokat külön maszkkal (valid\_fringing\_mask) összegyűjti és egy új maszkot (contour\_mask) készít. A kontúrok vizsgálatával a függvény segít elkerülni a nagy homogén lila területek téves észlelését.

A függvény a detektálás végén egy igaz vagy hamis érték mellett a hibás területeket tartalmazó bináris maszkot adja vissza, amennyiben nem található purple fringe a képen akkor csak fekete képet ad.

Képalkotási hibák javítása

A hibák javítását végző függvények a transformations.py fájlban találhatóak meg. A függvények meghívásának sorrendje is nagyon fontos a javítás során, mivel a nem megfelelő sorrend felállításának hiánya befolyásolhatja a javított kép minőségét. A sorrend a következő:

1. Elmosódottság javítása: Egyes javító algoritmusok például kontraszt felerősíthetik az elmosódást, így érdemes először enyhíteni.
2. Expozíciós hibák javítása: Az elmosódás nélküli képek hisztogramja pontosabb, ezért került a második helyre.
3. Purple fringe javítása: A színtorzulások színtérfüggők és érzékenyek a helytelen expozícióra.
4. Vörös szem hiba javítása: Az esetleges színtorzulások és expozíció javítása után érdemes elvégezni, mivel a korrigált, tisztább színekkel rendelkező kép stabilabb eredményt biztosít.

Elmosódottság javítása:

Az elmosódott képek javítását a program a blur\_correction és a hozzá tartozó alfüggvények segítségével végzi. A rendszer az elmosódás típusa alapján egy neurális háló segítségével megbecsüli az adott képhez tartozó Point Spread Function (PSF) kernel alakját, majd a becsült kernel alapján Wiener-dekonvolúcióval visszaállítja az eredeti kép élesebb változatát.

Működése:

A képek javítása két fő lépésből áll:

1. PSF predikció: egy előre betanított neurális hálózat a kép alapján becslést ad az elmosódást okozó kernelre.
2. Wiener dekonvolúció: a becsült kernel segítségével visszafejtjük az elmosódást a képen.

Ez a kombináció lehetővé teszi az elmosódással rendelkező képek automatikus, típusfüggő javítását.

PSF prediktor neurális háló

Két különböző PSF prediktor modellt használ a program. Mindkettő egy-egy elmosódás típus kezelésére. A modellek azonos architektúrával rendelkeznek, de eltérő típusú képeken lett tanítva.

A hálózat célja, hogy egy bemeneti RGB képből megbecsülje az elmosódást okozó 15x15-ös PSF kernel értékeit. A bemenetet először 128x128 pixelesre méretezzük. A neurális háló első részét egy egyszerű konvolúciós encoder alkotja, amely két egymás utáni konvolúciós rétegből áll. A első réteg 32 darab szűrőt alkalmaz, a második pedig 64-et, mindkettő után ReLU aktiváció következik, amely a teljes térbeli dimenziót egyetlen értékre tömöríti csatornánként 1x1 méretre, így kivonva a globális jellemzőket.

A kinyert 64 jellemzőt egy teljesen összekapcsolt réteg dolgozza fel, amely 225 kimentet állít elő, ami megfelel a 15x15 méretű kernel összes elemének. Ezután az eredményt egy 4D tenzorrá alakítjuk és ReLU aktivációval biztosítjuk, hogy a kernel elemei ne legyenek negatívak. Végül normalizálás történik úgy, hogy a kernel elemeinek összege 1 legyen. Ez biztosítja, hogy az eredmény egy valószínűségi eloszlásnak megfelelő súlymátrixként funkcionáljon, amelyet a dekonvolúció során közvetlenül fel lehet használni.

A tanításhoz használt adatforrás DIV2K képgyűjtemény speciális előkészített változatán történt, amely minden elmosódott képhez tartalmazott egy .npy formátumú kernelmátrixot is. A tanítóképeket RGB formátumban töltjük be és egységes pixelméretre méretezzük, hogy illeszkedjenek a modell bementéhez.

A háló tanítása során a cél az volt, hogy a bemeneti képből regresszióval előállítsuk a hozzá tartozó valós PSF kernelmintát. A tanítási célfüggvényként a közönséges négyzetes hiba (MSE) került alkalmazásra, amely a prediktált és a valós kernel közötti eltérést méri. A hálót Adam optimalizálóval tanítottuk, 1e-4 tanulási rátával, 20 epochon keresztül. A tanulás végén a legjobb modell súlyait menti el a program, hogy azokat később inferáláskor újra betölthető legyen.

Wiener dekonvolúció működése

Miután a hálóval becsültük az adott elmosódott képhez tartozó PSF kernel alakját, a kép helyreállítása Wiener-dekonvolúcióval történik. Ez a módszer frekvenciatartományban működik: először a bemeneti kép minden színcsatornáját külön-külön Fourier-transzformációnak vetjük alá, ugyanúgy, mint az elmosódást okozó kernelmátrixot. A dekonvolúciós lépés lényege, hogy a kernel komplex konjugáltját szorozzuk a kép spektrumával, majd elosztjuk a kernel spektrumának abszolút négyzetével, amelyhez egy kis értékű állandót (K) is adunk a nevezőhöz a stabilitás érdekében. Ez az állandó (tipikusan 0.01) segít elnyomni a spektrum zajos részeit.

A dekonvolúció után a kapott frekvenciatartománybeli eredményt inverz Fourier-transzformációval alakítjuk vissza a térbeli tartományba. A végső eredményt klippeljük (0 és 1 közé), majd 8 bites formátumba alakítva visszakódoljuk RGB képpé. Ez a folyamat lehetővé teszi, hogy az eredetihez hasonló, de jóval élesebb képet állítsunk elő az elmosódott bemenet alapján.

A fő függvény a folyamat végén visszaadja a javítás során elkészült élesebb képet amit a felhasználó először a grafikus felületen keresztül tekinthet meg.

Expozíciós hibák javítása:

Az alul- és túlexponált képek javítása exposure\_correction függvény és a hozzá tartozó alfüggvények használatával történik. Az algoritmus kontrasztkorlátozott adaptív hisztogram kiegyenlítést (CLAHE) és gamma korrekciót kombinál, hogy a kép általános kontrasztját és fényerejét kiegyensúlyozza, miközben a jól exponált részek nincsenek túlerősítve.

Működés:

1. Fényerő becslése:

Az algoritmus először meghatározza a kép átlagos fényerejét a HSV színtér V csatornájának átlaga alapján, Ez a mérték segít meghatározni, hogy a kép erősen-, enyhén alulexponált, normál fényerőjű vagy túlexponált.

1. Gamma korrekció

A gamma korrekció segít az általános fényerő korrekciójában. Az algoritmus automatikusan eldönti, hogy alkalmazza-e a gamma korrekciót:

* Erősen alulexponált kép esetén: gamma = 1.5
* Közepesen alulexponált kép esetén: gamma = 1.2
* Túlexponált kép esetén: nem alkalmaz gamma korrekciót

A gamma korrekció csak a V-csatornára kerül alkalmazásra, amely a fényerőt képviseli.

1. CLAHE alkalmazása

Az algoritmus lokálisan javítja a kontrasztot az árnyalat megőrzése mellett. A működés során a teljes V-csatorna több kis csempére oszlik, és mindegyiken külön kontrasztjavítás történik, elkerülve a túlillesztést.

A CLAHE hatásának mértéke a cliplimit paraméteren keresztül szabályozható, amelyet a fényerő értéke alapján dinamikusan állítunk be. Ha a fényerő értéke alacsonyabb mint 70, akkor erőteljesebb kontrasztjavítást alkalmazunk magasabb cliplimit értékkel (4.0). Enyhén alulexponált képeknél (70 és 120 közötti fényerő) közepes értéket (3.0) használunk. Amennyibben a kép fényereje a túlexponált tartományba esik (160-200), az érték visszafogottabb értékre (2.0) csökken, míg erősen túlexponált képeknél (200 felett) a legkisebb értéket (1.5) alkalmazzuk, hogy elkerüljük a világos területek további túlhangsúlyozását.

1. Színmegőrzés

Annak érdekében, hogy a korrekció után a színek ne tűnjenek mesterségesnek vagy túltelítettnek, az algoritmus a gamma korrekció és a CLAHE által módosított V-csatornát átlagolja. Ez kiegyensúlyozottabb és természetesebb megjelenést biztosít.

A módosított V-csatorna ezután visszakerül az eredeti HSV színtérbe, majd BGR formátumba konvertálva előáll a javított kép.

Piros szem hibák javítása:

A piros szem hibákat tartalmazó képek javítása a red\_eye\_correction függvény segítségével történik. Az algoritmus a bemenetként kapott színes BGR képet és a detektálás során megállapított szemrégiók koordinátáit úgynevezett bouding boxokat használja a vörös effektek korrigálására, amely a következő lépésekben történik:

1. Szemrégió kivágása és színcsatornákra bontása

A szem területe kiemelésre kerül a teljes képből, majd szétbontjuk kék, zöld és piros csatornákra.

1. Maszk készítése a piros szem detektálásához

A piros szem régiókat olyan pixelekként azonosítjuk, ahol a piros színcsatorna legalább 20 egységgel meghaladja a kék és zöld csatornák együttes értékét és emellett a piros csatorna értéke önmagában is nagyobb mint 80. Ez a kettő feltétel biztosítja, hogy csak a valóban erőteljesen piros területek kerüljenek kijelölésre, kiszűrve a halványabb vagy természetes vörös tónusokat.

1. Kontúrok kinyerése és maszkolás

A maszk kép kontúrjai kinyerésre kerülnek, majd kiválasztjuk a legnagyobb területű kontúrt, amely feltételezhetően a piros pupillának felel meg. Ezután mofológiai műveletek zárás és dilatáció segítenek a lyukak betömésében, és egy folytonos maszkrégió előállításában.

1. Színkorrekció átlagolt textúrával

A piros szín eltávolításához a kék és zöld csatornák átlaga kerül felhasználásra, amelyet a maszk által kijelölt területeken átmásolunk a piros pixelek helyére. A cél nem csupán a piros pixelek színének megváltoztatása, hanem a környező szemtextúra és színárnyalat megtartása is. A korrekciót a kép színes formájára alkalmazzuk, és visszaillesztjük a kép eredeti szemrégiójába.

Purple fringe hiba javítása

A lila elszíneződés javítása a purple\_fringe\_correction függvény segítségével történik. A feldolgozás egy előzetesen detektált fringing maszk alapján történik, amely jelöli a hibás, lilás árnyalatú régiókat. Az algoritmus első lépésként morfológiai módszerekkel tisztítja a maszkot, majd az elszíneződés súlyosságától függően különböző javítási stratégiákat alkalmaz. A cél a hiba eltüntetése a lehető legkevesebb szín és részletveszteség mellett.

Működés

1. Maszk tisztítása

A javítás előtt a maszkot morfológiai műveletekkel tisztítjuk. Először zárást alkalmazunk, amellyel eltüntetjük a maszkban található apró lyukakat és megszüntetjük a szakadásokat. Ezt követően nyitási műveletre kerül sor, amely segít eltávolítani a kisebb, izolált zajos foltokat. Ez a tisztítási lépés biztosítja, hogy a kijelölt lila területek összefüggőbbé és pontosabbá váljanak, így a későbbi javítási műveletek megbízhatóbban hajthatók végre.

1. Elszíneződés arányának mérése

A maszkban szereplő fehér pixelek arányából határozza meg a program, hogy a kép mekkora részét érinti a lila elszíneződés. Ez határozza meg, milyen javítási stratégia kerül alkalmazásra.

1. Javítási stratégiák

A lila elszíneződés javítása a hiba mértékéhez igazodik, amelyet a maszkon szereplő hibás pixelek aránya alapján határozunk meg. Három eltérő javítási módszert alkalmazunk, attól függően, hogy az elszíneződés kis, közepes vagy nagy kiterjedésű.

Ha az érintett területek aránya kisebb, mint öt százalék, akkor precíz, iteratív halványítási technikát alkalmazunk a HSV színtérben. Ennek során a maszk által kijelölt pixelek színárnyalatát, telítettségét és fényerejét fokozatosan csökkentjük. Az első iterációban 0.95-szörös csökkentést végzünk, amelyet minden körben tovább mérséklünk 0.98-os szorzóval. Minden módosítás után újra lefuttatjuk a detektálást, és ha a maszk már nem változik, vagy három egymást követő alkalommal nincs változás, a folyamat leáll. Ez a módszer hatékonyan halványítja el a nem kívánt lilás elszíneződést, miközben megtartja a kép színvilágát.

Amennyibben a lila területek aránya öt és húsz százalék közé esik, egy 5x5-ös medián szűrőt alkalmazunk a teljes képre. Ezután kizárólag a maszkkal kijelölt régiókat cseréljük ki a mediánnal simított változatra. Ez a megközelítés megőrzi a kép többi részének a részletgazdagságát, miközben finoman korrigálja a elszíneződési hibákat.

Ha az elszíneződés aránya meghaladja a húsz százalékot, akkor enyhe Gauss szűrőt alkalmazunk a teljes képre, 5x5-ös maszk és nullás szigmaérték mellett. A Gauss szűrés segítségével a program célja, hogy a hiba területén lágyabb átmeneteket hozzon létra, ezzel csökkentve az elszíneződés vizuális hatását anélkül, hogy a kép struktúrája jelentősen sérülne.

Osztályozás

A program a hibadetektálás és a képi hibák javítását követően lehetőséget biztosít a képek mentésére. A mentés során a felhasználó dönti el, hogy az eredeti vagy a javított képváltozatott szeretné elmenteni. A mentési lépés kézzel indítható gombbal a grafikus felületen keresztül. Ezen felül a mentés automatikus osztályozást is végez a képen látható tartalom alapján egy előre betanított YOLOv8 modell segítségével. A rendszer a képeket kategóriák szerint alkönyvtárakba rendezi, megkönnyítve azok további felhasználását.

Automatikus osztályozás YOLOv8 modellel

A képek osztályozása a classify függvényben történik, amely egy betanított YOLOv8 modellt használ. A modell az adott képre vonatkozó legnagyobb valószínűségi osztályt határozza meg, majd a hozzárendelt főkategória szerint alkönyvtárba sorolja a képet.

Az osztályok a programban előre meghatározott osztályok és azok főosztályba csoportosítása alapján működnek. A fő kategóriák a következők: Flowers, Cars, Landscapes, Animals valmint Group of People, melyek a predikció eredményétől függően mappákba rendezik a képeket.

A modell tanítása az images.cv oldalon található képekből készült adatbázissal történt. A képek minden osztályhoz külön mappában helyezkedtek el és a YOLOv8 tanításhoz szükséges annotációkat .txt fájlok formájában, YOLO formátumban generáltuk.

A data.yaml fájl tartalmazza az osztályok számát és az images.cv alapján létrehozott osztályneveket a names listában. A tanítás során biztosítottuk, hogy az osztályozási struktúra teljes mértékben megfeleljen a fenti osztályoknak.

A classify függvény a prediktált osztály alapján kiválasztja a megfelelő fő kategóriát, és a kép automatikusan a megfelelő alkönyvtárba kerül mentésre.

Tesztelés