2024. DECEMBER 2.

MESTERSÉGES INTELLIGENCIA BEADANDÓ

SZÁMÍTÓGÉPES LÁTÁS ALAPÚ OBJEKTUM DETEKTÁLÁS

TARJÁNYI DÁNIEL;SZŰCS KRISZTIÁN

Tartalom

Bevezetés

A projekt célja egy mesterséges intelligencia alapú alkalmazás fejlesztése, amely képes valós idejű objektumdetektálásra videókban. Az alkalmazás lehetőséget ad a felhasználónak arra, hogy saját videót töltsön fel, választhassa az objektumok típusát, valamint dönthet a feldolgozási eszköz (CPU vagy GPU) használatáról. A projekt során a **Streamlit** könyvtárat használtuk a grafikus felület létrehozásához, az **Ultralytics YOLO** modellt az objektumok detektálására, és az **OpenCV**-t a videók feldolgozásához. Az alábbiakban részletesen bemutatjuk a projekt megvalósításának folyamatát, az általunk tapasztalt nehézségeket, és azokat a tanulságokat, amelyeket a projekt során nyertünk.

A projekt megvalósítása

Technológiák és eszközök

- **Streamlit**: Könnyen használható Python-alapú keretrendszer interaktív webes alkalmazások készítéséhez.
- **Ultralytics YOLO**: Az objektumdetektálás legnépszerűbb modellje, amely gyors és pontos detektálást tesz lehetővé.
- **OpenCV**: A számítógépes látás alapkönyvtára, amelyet videók feldolgozására és mentésére használtunk.
- **PyTorch**: A YOLO modell futtatására GPU-támogatással.

Funkciók

1. **Videófeltöltés**: A felhasználók feltölthetik saját videóikat, amelyekből az alkalmazás automatikusan feldolgozási formátumot készít.
2. **Objektumosztályok kiválasztása**: A felhasználók meghatározhatják, hogy mely objektumokat szeretnék detektálni (pl. autók, gyalogosok, kerékpárok).
3. **Feldolgozási eszköz kiválasztása**: A felhasználó választhat a CPU vagy GPU között, attól függően, hogy milyen hardver áll rendelkezésre.
4. **Videófeldolgozás valós időben**: A feltöltött videók minden egyes képkockáját feldolgozzuk, és a detektált objektumokat megjelenítjük.
5. **Kimeneti videó letöltése**: A feldolgozott videót a felhasználó letöltheti.

Nehézségek és megoldások

GPU és CUDA támogatás

Probléma: A GPU nem jelent meg automatikusan elérhető eszközként a választási lehetőségek között, annak ellenére, hogy a nvidia-smi parancs szerint a GPU elérhető volt.

Ok: A probléma oka az volt, hogy a PyTorch nem GPU-kompatibilis verziója volt telepítve.

Megoldás:

- Telepítettük a GPU-kompatibilis PyTorch verziót a következő paranccsal:

```
pip install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu118
```

- Ellenőriztük, hogy a CUDA támogatás elérhető-e

```
import torch

print(torch.cuda.is_available())

print(torch.cuda.get_device_name(0))
```

Ezután a GPU opció sikeresen megjelent az alkalmazásban.

Videók feldolgozásának sebessége

Probléma: Nagyobb felbontású videók feldolgozása CPU-val lassú volt, ami rontotta a felhasználói élményt.

Megoldás: A GPU-t használtuk a YOLO modell gyorsítására, amely a következő módosítással érhető el:

```
self.modell = YOLO(modell_utvonal).to("cuda")
```

Ha a GPU nem volt elérhető, az alkalmazás automatikusan CPU-ra váltott:

```
eszkoz = "cuda" if torch.cuda.is_available() else "cpu"
```

Fájlkezelés és videómegjelenítés

Probléma: A feldolgozott videók időnként nem jelentek meg a Streamlit felületén.

Ok: A videó mentésére és megjelenítésére használt OpenCV és Streamlit között néha formátumkompatibilitási problémák merültek fel.

Megoldás: Az OpenCV mentési beállításait finomhangoltuk, és a Streamlit videómegjelenítőjét közvetlen fájlolvasással használtuk:

```
with open(kimeneti_utvonal, "rb") as video_file:
    video_bytes = video_file.read()
    st.video(video_bytes)
```

Objektumosztályok kezelése

Probléma: Az összes objektumosztály egyszerre történő feldolgozása túlzott terhelést jelentett a rendszer számára.

Megoldás: A felhasználók számára lehetővé tettük, hogy csak azokat az objektumokat válasszák ki, amelyekre szükségük van. Ezt egy Streamlit multiselect widgettel oldottuk meg:

```
kivalasztott_osztalyok = st.sidebar.multiselect(
    "Objektumok kiválasztása",
    options=list(OSZTALY_NEVEK.values()),
    default=list(OSZTALY_NEVEK.values())
)
```

Tanulságok

1. **Hardverfüggetlen tervezés fontossága**: A GPU és CPU támogatásának dinamikus kezelése jelentősen javította az alkalmazás rugalmasságát.

2. **Optimalizáció szükségessége**: Az OpenCV és YOLO működésének finomhangolása elengedhetetlen volt a nagyobb videók hatékony feldolgozásához.
3. **Felhasználói élmény előtérbe helyezése**: Az interaktív beállítások, például az objektumok kiválasztása és a feldolgozási mód megválasztása, pozitív visszajelzéseket eredményezett.
4. **Debugging fontossága**: A CUDA kompatibilitási problémák megoldása mélyebb megértést adott a PyTorch és NVIDIA eszközök működéséről.

Jövőbeli fejlesztési lehetőségek

Valós idejű kamera támogatása: Az alkalmazás kiterjesztése élő kameraképekre, például webkamerákra.

Több modell támogatása: A YOLO mellett más objektumdetektáló modellek integrációja, például SSD vagy Faster R-CNN.

További formátumok támogatása: Az alkalmazás kompatibilitásának növelése a különböző videóformátumokkal.

Automatikus optimalizáció: Az alkalmazás automatikusan választhatná a legjobb feldolgozási módot (CPU/GPU) a videó mérete és a rendelkezésre álló hardver alapján.

Következtetés

A projekt során létrehoztunk egy modern, interaktív objektumdetektáló alkalmazást, amelyet a felhasználók könnyen használhatnak különböző célokra. Az alkalmazás mind CPU, mind GPU támogatással működik, és testreszabható beállításokat kínál. Bár számos kihívással szembesültünk, a problémák megoldása révén értékes tapasztalatokat szereztünk a mesterséges intelligencia alkalmazásfejlesztésében. A jövőbeli fejlesztések tovább növelhetik az alkalmazás hatékonyságát és felhasználói élményét.