Dokumentace k závaznému výsledku projektu

Projekt:

Inteligentní robotická ochrana zdraví ekosystému hydroponického skleníku

FW01010381

Výstup:

Software pro detekci škůdců a chorob na listech

Tento projekt je spolufinancován se státní podporou Technologické agentury ČR a Ministerstva průmyslu a obchodu v rámci **Programu TREND.**

T A
Č R

www.tacr.cz www.mpo.cz

Obsah

Představení software

Pro sledování stavu porostu ve skleníku je důležitá informace o přítomných škůdcích či jiných poškození rostlin. Tento software umožnuje nahrát fotografii pořízenou přímo ve skleníku v porostu. Fotografie je zanalyzována a v případě výskytu nějakého poškození, je určen jeho druh. Jsou rozpoznávány následující vady porostu:

- výskyt molice skleníkové,
- poškození chorobou padlí,
- poškození vrtalkami,
- poškození žírem housenek motýla Tuta absoluta,
- poškození žírem housenek bez určení přesného druhu.

Ukázka vstupního snímku pro vyhodnocení typu poškození porostu je na snímcích níže.

(a) Výskyt Molice skleníkové

(b) Poškození chorobou padlí

Po vyhodnocení softwarem lze v případě pozitivního nálezu aplikovat vhodné opatření dle typu a rozsahu poškození / napadení škůdcem:

1) ***Molice skleníková*** je klíčovým škůdcem na rajčatech ve skleníku a vyskytuje se prakticky vždy. Při zanedbané ochraně dokáže tento škůdce způsobit zásadní ekonomické škody na porostu. Základním prostředkem pro monitoring molice jsou žluté lepové desky. Detailnější popis se speciálně vyvíjenou aplikací pro detekci a počítání množství molic na žlutých lepových deskách je v kapitole 3.4. Na nich se počítají zachycené molice jednou týdně. Přibližně do v průměru 20 jedinců/žlutá lepová deska lze postupovat proti molicím pouze biologickou ochranou. Nad tuto hranici je obvykle vhodné použít korekční postřik vybraným insekticidem. Detekce larev či dospělců molic na listech lze považovat za vhodný doplňkový nástroj. Především při odhalování ohnisek výskytu takzvaných hot spotů. Díky tomu je možné ochranu proti molicím lépe organizovat a vyšší dávku biologické ochrany či postřiky směřovat selektivně přímo do těchto ohnisek.
2) ***Padlí*** je obligátní parazitická houba, která se na listech rajčete projevuje tvorbou bílých povlaků, které jsou tvořeny myceliem. Obvykle se padlí ve skleníku objevuje buď na velmi citlivých odrůdách rajčat nebo při rozkolísání vnitřního klimatu. Je-li padlí zjištěno včas na počátku vývoje je možné vhodnými opatřeními dosáhnout jeho vymizení. Mezi ta patří

úprava klimatu popřípadě aplikace fungicidů na bázi síry. Tedy rozpoznání padlí v ranné fázi vývoje na listech je důležitý předpoklad pro zvládnutí ochrany bez nutnosti zapojení intenzivního fungicidního programu.

3) ***Vrtalky*** patří spíše mezi méně významné škůdce. Larva tvoří typické chodbičky v listech (miny). Proto, aby jejich poškození mělo reálný vliv na úroveň výnosu muselo by dojít k opravdu kalamitnímu přemnožení. Tedy detekce vrtalek na listech obvykle neznamená nutnost okamžitého zásahu. Avšak je dobré její výskyt průběžně sledovat a případně nasadit vhodnou biologickou ochranu.
4) ***Tuta absoluta*** – před poškozením listů motýlem Tuta absoluta by měl předcházet jeho záchyt ve feromonových lapačích. Tedy agronom obvykle díky záchytu dospělců v lapači může počítat s tím, že dojde k poškození porostu housenkami. Je-li pomocí obrazové detekce indikováno možné poškození motýlem T. absoluta je s ohledem na jeho škodlivost nutné provést ošetření vhodným larvicidním přípravkem. Zároveň s ohledem na to, že poškození je podobné jako v případě vrtalek, je vhodné vyloučit tuto případnou záměnu.
5) Poškození ***žírem housenkami*** na listové ploše je obvykle indikace k ošetření vybraným larvicidem. Plošná analýza obrazových dat z celého skleníku může pomoci odhalit rozsah tohoto napadení. Na základě toho se pak rozhoduje o aplikaci jen do vybrané oblasti nebo v celém skleníku.
6) ***Cladosporium fulvum (černˇ rajčatová)*** je choroba, která se projevuje tvorbou tmavého/černého mycelia na povrchu listu. Dochází tak ke tvorbě černých skvrn různé velikosti. Jde o problematičtější chorobu než padlí a její výskyt je obvykle možné jen omezovat. Analýza obrazových data může přispět k časné detekci a přispět tak k celkově lepšímu výsledku v managementu ochrany rostlin.

Uživatelská příručka

Aplikace je tvořena sadou virtualizačních kontejnerů technologie Docker. Pro její provozování je potřebné na dané platformě již funkční virtualizační prostředí Docker včetně rozšíření Docker-compose. Software je tvořen rozhraním REST API umožňujícím asynchronní zpracování požadovaných snímků včetně jednoduchého testovacího formuláře.

Spuštění a přístup k API

Spuštění veškerých interních součástí software se provádí pomocí jednoho příkazu:

docker compose up -d

POZOR prvotní spuštění může trvat až desítky minut – doba závisí na kvalitě internetového připojení.

Po spuštění je API dostupné na adrese:

http://IP_ADDESS:5001/api

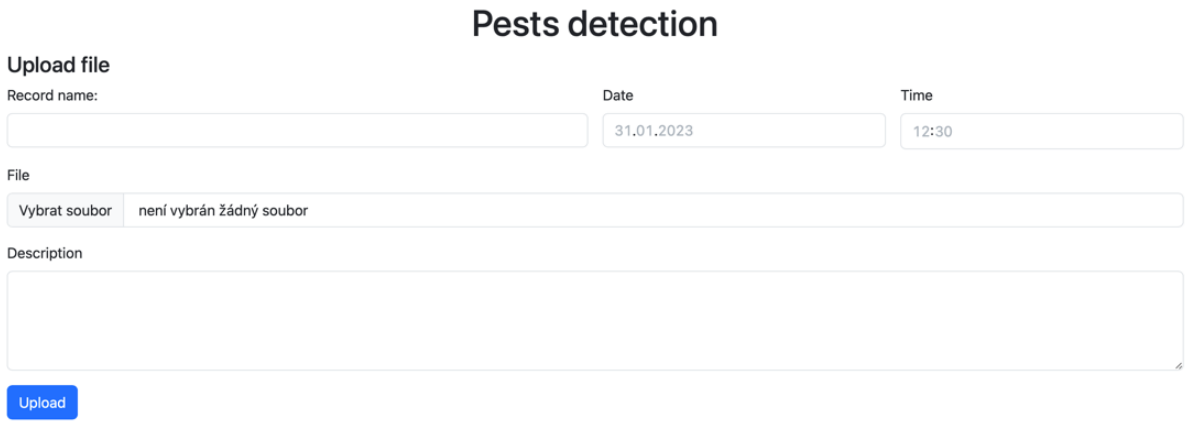
Testovací formulář je dostupný na adrese:

http://IP_ADDRESS:5001/web/upload

kde IP_ADDRESS reprezentuje jednoznačnou IP adresu serveru/počítače na kterém je SW spuštěn.

Testovací formulář

Pro otestování funkčnosti SW je dostupný testovací formulář. Na níže uvedeném obrázku je uveden snímek tohoto formuláře:



S využitím formuláře je možné zaslat požadovaný snímek na zpracování pomocí dostupného API. Formulář obsahuje veškeré pole, které je nutné při zadávání požadavku na zpracování uvést. Tj. název záznamu, datu a čas pořízení a popis záznamu. Dále je také nutné vybrat

soubor, který bude odeslán ke zpracování. **Pro správnou funkčnost formuláře je nutné vyplnit všechny položky!**

Odesláním formuláře se s využitím API provede nahrání snímku pro zpracování a zadání úlohy zpracování. Výsledkem odeslání formuláře je tedy aktivní úloha. Uživateli je navrácen záznam ve formátu JSON ve stejné formě, jak jej vrací přímo API (viz. následující kapitola).

API rozhraní

API rozhraní aplikace je tvořeno jednotlivými koncovými body:

Koncový bod:

http://IP_ADDESS:5001**/api/stages**

Použitá http metoda:

- GET

Parametry:

- Žádné

Návratová hodnota:

- JSON obsahující jednotlivé dostupné stavy

Příklad výstupu:

```
[
  {
    "ID": 4,
    "name": "preprocessing"
  },
  {
    "ID": 3,
    "name": "processed"
  },
  {
    "ID": 2,
    "name": "processing"
  },
  {
    "ID": 1,
    "name": "upload"
  }
]
```

Popis:

Jednotlivé zadané úlohy se mohou nacházet v několika stavech. Stav „upload“ nastane okamžitě po zadání úlohy a znamená, že požadovaný snímek je připraven na zpracování. Stavy „preprocessing“ a „processing“ indikují, že daná úloha již začala být zpracována, nicméně její zpracování ještě není dokončeno. Stav „processed“ indikuje, že daná úloha již byla zpracována a jsou dostupné výsledky. Tento koncový bod vrátí ve formě JSON výstupu všechny dostupné stavy a jim odpovídající interní ID.

Koncový bod:

http://IP_ADDESS:5001**/api/records**

Použitá http metoda:

- GET

Parametry:

- Žádné

Návratová hodnota:

- JSON obsahující všechny doposud zadané úlohy

Příklad výstupu:

```
[
  {
    "ID": 1,
    "filename": "1_2023-05-31_181600.jpg",
    "name": "Task 1",
    "result": // zkráceno ,
    "stage": 3,
    "stageName": "processed",
    "timestamp": "Wed, 31 May 2023 18:16:00 GMT"
  },
  {
    "ID": 2,
    "filename": "2_2023-06-16_152400.jpg",
    "name": "Task 2",
    "result": // zkráceno,
    "stage": 3,
    "stageName": "processed",
    "timestamp": "Fri, 16 Jun 2023 15:24:00 GMT"
  },
…
]
```

Popis:

Tento koncový bod vrátí všechny doposud zadané úlohy pro zpracování. Součástí výstupu jsou veškeré zadané informace o úloze, včetně stavu, ve kterém se nachází.
V případě, že je již úloha zpracována, je zobrazen i výsledek zpracování.
POZOR: V případě, že již bylo zadáno velké množství úloh, může být výstup velmi dlouhý.

Koncový bod:

http://IP_ADDESS:5001**/api/records** /<recordID>

Použitá http metoda:

- GET

Parametry:

- recordID – ID již existující úlohy

Návratová hodnota:

- JSON obsahující informace o již existující úloze

Příklad výstupu:

```
[
  {
    "ID": 1,
    "filename": "1_2023-05-31_181600.jpg",
```

```
    "name": "Task 1",
    "result": // zkráceno ,
    "stage": 3,
    "stageName": "processed",
    "timestamp": "Wed, 31 May 2023 18:16:00 GMT"
  }
]
```

Popis:

Tento koncový bod vrátí informace o specifické exitující úloze. Součástí výstupu jsou veškeré zadané informace o úloze, včetně stavu, ve kterém se nachází. V případě, že je již úloha zpracována, je zobrazen i výsledek zpracování.

Koncový bod:

http://IP_ADDESS:5001**/api/uploader**

Použitá http metoda:

- POST

Parametry:

- file – nahrávaný soubor
- recordName – název záznamu
- date - datum
- time - čas

Návratová hodnota:

- ID nově vytvořené úlohy

Popis:

Tento koncový bod slouží k nahrávání snímků určených pro zpracování. Jeho návratovou hodnotou je ID zadané úlohy, které může sloužit pro pro zjištění jejího stavu nebo pro načtení výsledku zpracování.

Popis výsledků klasifikace

Výsledná informace o výskytu poškození porostu je uložena v JSON souboru, který obsahuje název a id predikované třídy a skóre predikce určující jistotu modelu, že je dané predikce správná.

Příklad výstupu pro jednotlivý obrázek:

```
[
  {
      'pred_class': ' Tuta_absoluta ',
      'pred_label': 8,
       'pred_score': 0.9615796208381653
  }
]
```

Technická dokumentace

Požadavky na SW:

- Komunikační rozhraní v podobě univerzálního API
- Zpracování fotografií s minimálním rozlišením 5 MPx
- Vyhledání typických poškození na listu
- Vyhodnocení poškození s ohledem na známé příčiny
- Sestavení základního doporučení ohledně protiopatření
- Uchovávání historických záznamů z provedených kontrol
- Příprava na rozšíření o detekci s využitím různých obrazových spekter

Implementace funkčních částí

Popisovaný software byl implementován s využitím kontejnerové technologie Docker. Jednotlivé části byly s cílem umožnit asynchronní zpracování rozděleny do několika nezávislých služeb.

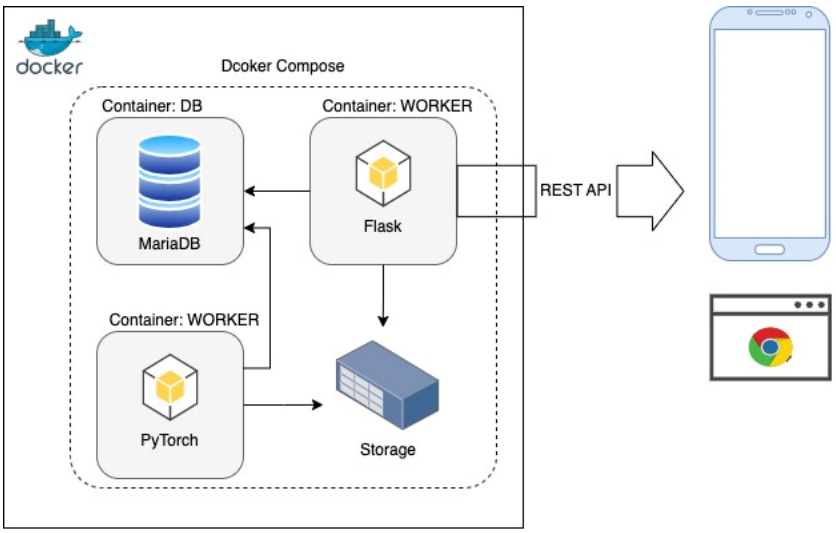
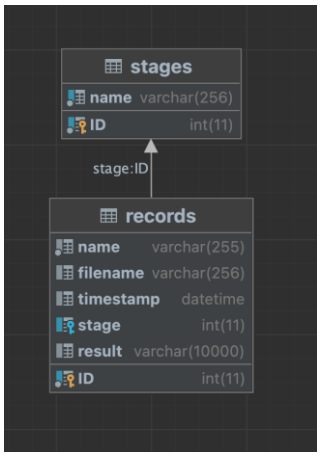


Diagram zobrazující propojení jednotlivých služeb

Databáze

Jako databázové pozadí byl vybrán databázový server MariaDB, který slouží ke komunikaci jednotlivých dalších komponent a společně se sdíleným uložištěm k uchovávání historických záznamů. Interní struktura databáze obsahuje dvě tabulky, jejichž vazba je znázorněna níže na obrázku:



Struktura databáze

Každá úloha zaslaná pro zpracování je vytvořena v podobě jednoho záznamu v tabulce „records“, který nese informace o úloze jako takové včetně výsledku zpracování.

API

Rozhraní aplikace je tvořeno jednoduchým REST API (viz. uživatelská příručka), které je implementováno s využitím programovacího jazyka Python a frameworku Flask. Toto rozhraní umožňuje zadávání jednotlivých úloh pro zpracování stejně jako získávání informací o jejich výsledku. Díky oddělení API do samostatného kontejneru je zaručena maximální rychlost odezvy.

Worker

Jedná se o samostatný kontejner implementující algoritmy pro zpracování obrazu. (viz. sekce Použité algoritmy a technologie). Tento kontejner komunikuje pouze s databází a pravidelně periodicky zjišťuje přítomnost nové úlohy pro zpracování. V případě, že je úloha pro zpracování dostupná, označí ji odpovídajícím stavem a začne zpracovávat. Díky tomuto oddělení probíhá zpracování asynchronně vzhledem k API. Dále je možné paralelní spuštění více instancí tohoto kontejneru a tím zvýšení propustnosti software.

Možnosti rozšíření

Vzhledem k použité architektuře je možné software dále rozšiřovat o další kroky předzpracování a zpracování informací. Rozšíření spočívá ve vytvoření odpovídajícího uzlu/kontejneru pro zpracování a jeho napojení na databázi a sdílené uložiště. Rozšířením stavů, kterých můžou jednotlivé úlohy nabývat, je možné průběžně sledovat jednotlivé fáze zpracování.

Použité algoritmy a technologie

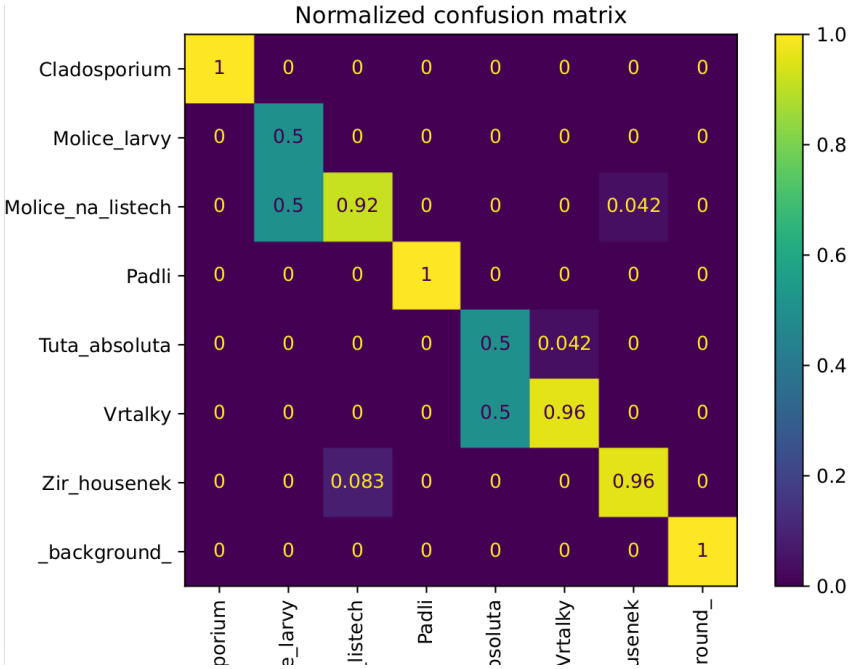
Software obsahuje funkci inference_engine, která se stará o inicializaci modelu, který je popsán konfiguračním .py souborem dle knihovny MMClassification[1] a natrénovanými váhami uloženými ve formátu .pth. V případě potřeby je možné model jednoduše zaměnit za jiný. Nyní se používá model sítě ResNet50, trénovaný se ztrátovou funkcí FocalLoss a rozměrech vstupu 678x678 pixelů. Dalším povinným vstupem je cesta k vstupnímu obrázku, který je třeba klasifikovat. Podporované formáty vstupních obrázků jsou následující: JPG, JPEG, BMP, PNG. Posledním vstupem je cesta k adresáři, kde jsou po spuštění programu uloženy výsledky predikce.
Program pracuje ve dvou krocích. Nejdříve dojde k inicializaci modelu a jeho přípravě pro predikci. V případě potřeby je možné upravit podrobnější podmínky predikce, například vybrat, zda výpočty budou probíhat na CPU či GPU. Poté probíhá samotná inference, jejímž výstupem je určení, zda je přítomno některé z poškození či se jedná o fotku zdravého porostu. Volitelně je možné na konci běhu programu vizualizovat vstupní obrázek s popiskem detekované třídy.
Po vyhodnocení softwarem lze v případě pozitivního nálezu aplikovat vhodné opatření dle typu a rozsahu poškození / napadení škůdcem popsaná v závěrečné zprávě projektu.

Popis výsledků

Výsledky detekce jsou uloženy do JSON souboru results.json, který obsahuje název a id predikované třídy a skóre predikce určující jistotu modelu, že je dané predikce správná.

Úspěšnost modelu byla otestována na testovací množině obrázků. Celkově bylo dosaženo přesnosti 0.87. Detaily záměn lze pozorovat v normalizované konfúzní matici níže, např. molice na listech byla zaměněna se žírem housenek nebo molice dospělci se svými larvami.

[1] Contributors, M. (2020). Openmmlab's image classification toolbox and benchmark.
URL: https://github. com/openmmlab/mmclassification.



Technická specifikace inference_engine

Parametry:

- konfigurační .py soubor dle knihovny MMClassification[2]
- Natrénované váhy ve formátu .pth
- vstupní obrázek, povolené formáty JPG, JPEG, BMP, PNG

Návratová hodnota:

- JSON obsahující výsledky predikce

Příklad výstupu pro jednotlivý obrázek:

```
[
  {
      'pred_class': ' Tuta_absoluta ',
      'pred_label': 8,
       'pred_score': 0.9615796208381653
  }
]
```

Popis:

Tato funkce slouží k predikci vstupního obrazu modelem. Výsledky detekce jsou uloženy do JSON souboru results.json, který obsahuje název a id predikované třídy a skóre predikce určující jistotu modelu, že je dané predikce správná.

[2] Contributors, M. (2020). Openmmlab's image classification toolbox and benchmark. URL: https://github. com/openmmlab/mmclassification.