Dokumentace k závaznému výsledku projektu

Projekt:

Inteligentní robotická ochrana zdraví ekosystému hydroponického skleníku

FW01010381

Výstup:

Software pro detekci molic na žlutých štítcích

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 vývoje populace polétavých škůdců v porostu se ve sklenících běžně používají lepové desky. Tito škůdci se na deskách zachytí a jsou následně ručně počítáni, což je pracný a zdlouhavý proces. Lidské počítání je také náchylné k chybám. Tento software umožnuje proces počítání škůdců zachycených na lepových štítcích zautomatizovat. Ukázka vizualizace výstupu je na obrázku níže.
V návaznosti na počet škůdců detekovaný na lepových deskách je vhodné zavést různá opatření (například postřiky či vypuštění vhodných predátorů pro kontrolu populace škůdců) pro předejití výraznějších škod na porostu.

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

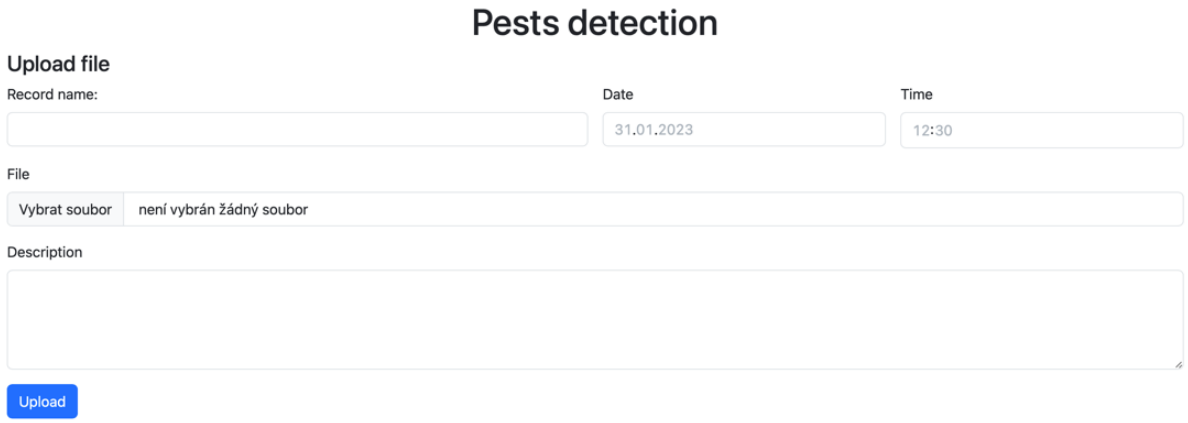
Testovací formulář je dostupný na adrese:

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 zjištění jejího stavu nebo pro načtení výsledku zpracování.

Popis výsledků detekce

Výsledná informace o počtu a pozici detekovaných moli je uložena v JSON souboru, který respektuje standartní COCO formát. Ten obsahuje především seznam zpracovaných obrázků, tj **images**, každý obrázek má přiděleno ID, file_name, rozměry. Dále je k dispozici seznam anotací, tj **annotations**, což jsou pozice jednotlivých detekovaných molic, každá anotace má vlastní ID, ID obrázku, ke kterému se váže, informaci o pozici zakódovanou jako souřadnice obdélníku (bbox) ve formátu x-levý, y-horní, šířka, výška, nakonec ještě skóre, které určuje jistotu modelu se kterou model tuto predikci udělal.

Příklad výstupního JSON souboru v COCO formátu:

```
[
  {
      'info': {},
      'licences': {},
      ' categories ': [
          {
                  'id ': 0,
                  'name ':  'whitefly '
```

```
                    }
                ],
        'images': [
            {
                    'id ': 0,
                    'file_name ':  'PA262255.jpg ',
                    'height ': 4608,
                    'width ': 3456,
                    },
            ...
                ],
        'annotations' : [
            {
                'image_id': 1,
                'bbox': [2056, 1816, 27, 28],
                'score': 0.9134126901626587,
                ' category_id ': 0,
                ' category_name ': whitefly,
                'segmentation': [],
                'iscrowd': 0,
                'area': 756,
                  },
            ...
                ],
  },
  . . .
]
```

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.

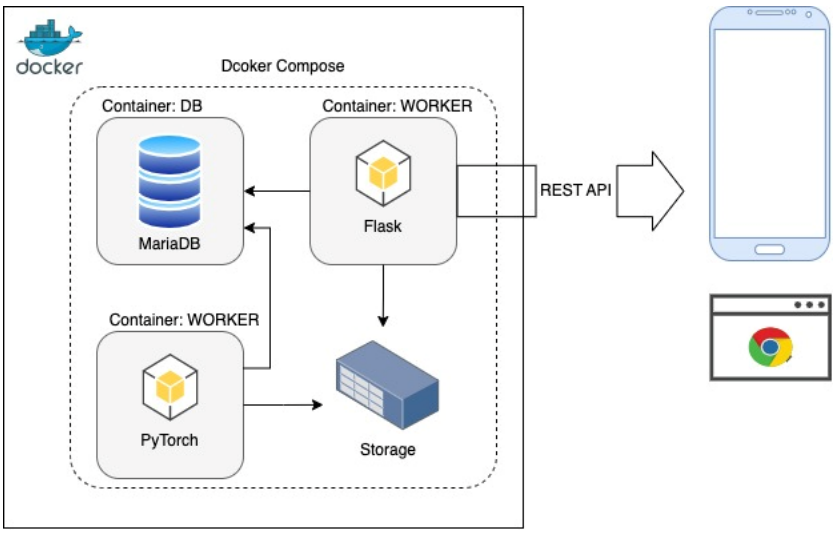
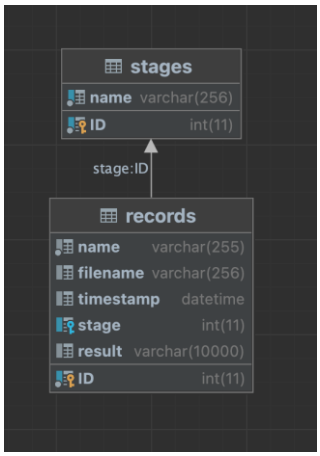


Diagram zobrazující propojení jednotlivých služeb

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 MMDetection[1] a natrénovanými váhami uloženými ve formátu .pth. V případě potřeby je možné model jednoduše zaměnit za jiný. Dalším povinným vstupem je cesta k vstupnímu obrázku, případně může být i zadána jen cesta k adresáři obsahujícímu více obrázků, v takovém případě budou predikovány všechna obrazová data ve složce. 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 třech krocích. V první fázi 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. Dále je možné volitelně uložit i vizualizaci výstupu. V tom případě bude ve výstupní složce vytvořen adresář visuals obsahující všechna vstupní obrazová data s vykreslenými detekcemi. Po inicializaci modelu program načte vstupní obrazová data a vytvoří JSON soubor test.json, kde jsou vstupní obrázky uloženy v COCO[2] formátu. Poté probíhá samotná inference, během které je vstupní obraz rozřezán na dílčí části, na kterých je provedena predikce. Výsledné detekce jsou následně vyfiltrovány a opět složeny dohromady.

Popis výsledků

Výsledné řešení implementuje TOODr50 model, trénovaný na unikátním datasetu vytvořeném v rámci projektu. Výsledné řešení dosahuje na testovací množině Precision 0.8378, Recall 0.8254 a F1-skóre 0.8314 (jedná se o harmonický průměr mezi recall a precision). Jsou to obdobné hodnoty, jako v prvotním testování dosahovala obsluha vs rostlinolékař, kdy výsledky obsluhy ve srovnání s rostlinolékařem dosáhly F1-skóre 0.81.

Výsledky detekce jsou uloženy do JSON souboru ve standardním COCO[3] formátu, kde jsou k dispozici pro případné další zpracování. Je tedy možné například jednoduše zjistit počet molic detekovaných na lepovém štítku i jejich přesnou pozici.

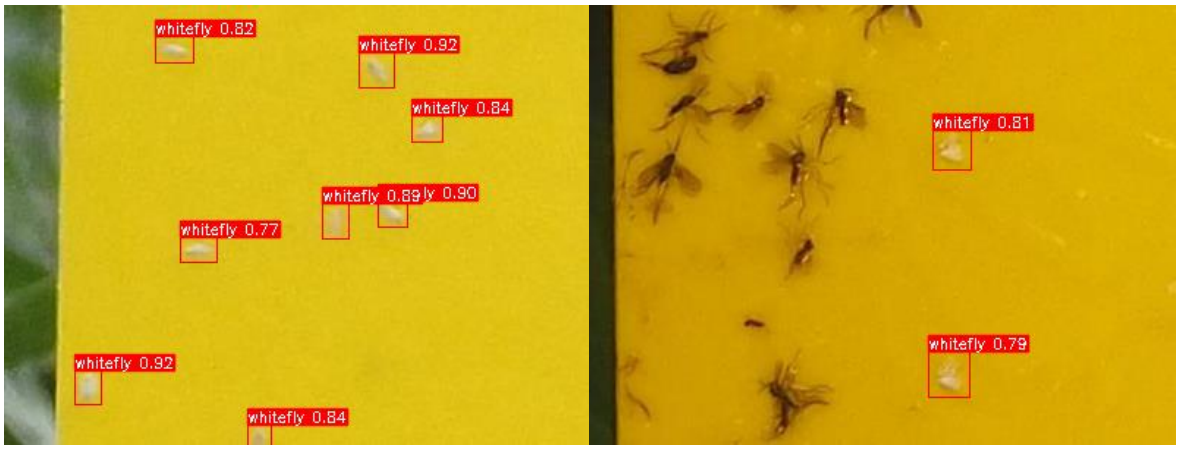
Ukázka vizualizace detekcí je na dvou výřezech obrázku níže.

---

[1] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., ... & Lin, D. (2019). MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155.

[2] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

[3] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.



Technická specifikace inference_engine

Parametry:

- konfigurační .py soubor dle knihovny MMDetection[4]
- Natrénované váhy ve formátu .pth
- vstupní obrázek, povolené formáty JPG, JPEG, BMP, PNG

Návratová hodnota:

- JSON obsahující výsledky predikce ve standartním COCO formátu
- JSON obsahující pouze výsledky predikce

Příklad výstupního JSON souboru v COCO formátu:

```
[
  {
      'info': {},
      'licences': {},
      ' categories ': [
          {
                  'id ': 0,
                  'name ':  'whitefly '
              }
          ],
      'images': [
         {
                  'id ': 0,
                  'file_name ':  'PA262255.jpg ',
                  'height ': 4608,
                  'width ': 3456,
                  },
      ...
              ],
```

[4] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., ... & Lin, D. (2019). MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155.

```
'annotations' : [
   {
                    'image_id': 1,
                    'bbox': [2056,1816,27,28],
                    'score': 0.9134126901626587,
                    'score': 0.9134126901626587,
                    ' category_id ': 0,
                    ' category_name ': whitefly,
                    'segmentation': [],
                    'iscrowd': 0,
                    'area': 756,
                      },
      ...
                ],
  },
  . . .
]
```

Příklad výstupu pro jednotlivý obrázek:

```
[
  {
      'image_id': 1,
      'bbox': [2056,1816,27,28],
      'score': 0.9134126901626587,
      'score': 0.9134126901626587,
      ' category_id ': 0,
      ' category_name ': whitefly,
      'segmentation': [],
      'iscrowd': 0,
      'area': 756,
  },
  . . .
]
```

Popis:

Tato funkce slouží k predikci vstupního obrazu modelem. Výsledky detekce jsou uloženy do JSON souboru ve standardním COCO[5] formátu, kde jsou k dispozici pro případné další zpracování. COCO JSON soubor obsahuje především seznam obrázků s přiděleným IMAGE_ID, dále seznam dostupných tříd s přiděleným CATEGORY_ID a konečně seznam anotací s přiděleným vlastním ID, pomocí IMAGE_ID a CATEGORY_ID navázaných ke kategorii a jednotlivým obrázkům.

JSON soubor “results.json” obsahuje pouze seznam anotací s přiděleným vlastním ID pomocí IMAGE_ID a CATEGORY_ID navázaných ke kategorii a jednotlivým obrázkům. Formát a obsah souboru “results.json” odpovídá poli 'annotations' ze standartního COCO dataset formátu.

[5] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

V obou případech je možné například jednoduše zjistit počet molic detekovaných na lepovém štítku.