Dumping Mapper

Andreas Kamilaris
Illya Averchenko
Job Römer
Onen Ege Solak
Wishal M Sri Rangan
Mian Tashfeen Shahid

Outline

1. Problem Statement
2. Dumping Mapper
3. Data Preparation
4. Detection Model
5. Classification Model
6. Back-end
7. Front-end
8. Questions

Problem Statement

- Illegal dumping has negative effects for many countries
  - Environmental risks
    - Contaminated water and soil
  - Human Health risks
    - Safety of food
    - Spread of disease (rodents)
  - Tourism
- Detect and clean dumping timely!
- Solution: a service that streamlines this process!

Dumping Mapper

- Two deep learning models
  - Detect waste on satellite images
  - Classify waste materials
- Web application dashboard
  - Show dumping sites and statistics on dashboard

Data Preparation

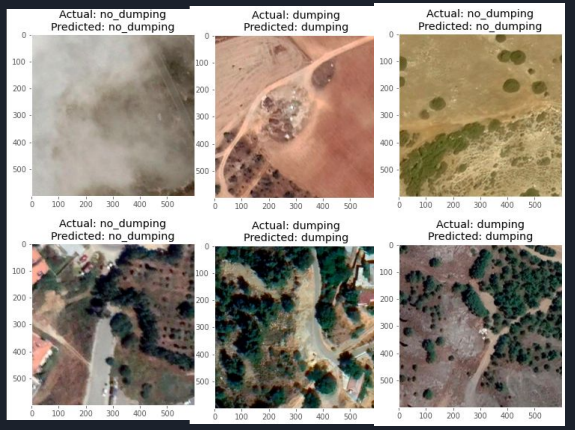
by Job Römer

- Annotation of satellite images (Detection)
  - Sort pictures on dumping occurrence
  - Bounding Boxes (Unused)
- Creation of synthetic data (Classification)
  - OpenCV (top) vs Blender (bottom)
  - Blender creates more realistic images
- Future Work
  - Reintroduce verification process
  - Better annotation of satellite images (delegate)
  - More realistic images (difficulties with Blender)
    - More diverse Blender models and materials
    - Fix background skewing issues
    - Fix coordinate dataset issues

Detection Model

by Illya Averchenko

- Model selection
  - CNN (VGG16)
- Pipeline configuration
  - 600x600px segments
  - Dataset split and augmentation
  - Output to the classification model
- Environment and libraries set up
  - Google Collab Pro
  - Object Detection API
  - TensorFlow.Keras
- Model Training
  - Accuracy
  - F1-score
- Results
- Improvement



Accuracy **96%**

F1-score **0.96**

Model details

VGG16
5 blocks
20 epochs
Augmented data
Dataset 1,000 images

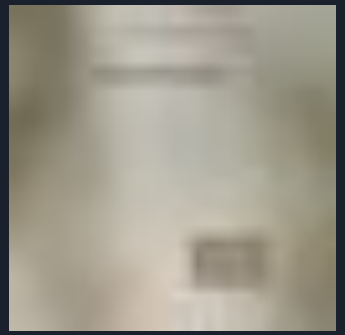
	precision	recall	f1-score	support
0	0.96	0.99	0.97	76
1	0.97	0.92	0.95	38
accuracy			0.96	114
macro avg	0.97	0.95	0.96	114
weighted avg	0.97	0.96	0.96	114

Classification Model

by Tashfeen Shahid Anwar

- The classification model uses synthetic data to train and test.
- After training the classification model predictions are done based on the classes of trash we are using.
- When We have a good accuracy on Synthetic Data we can test on real data.
- Our future plan is to give in a picture which has mixed dumping (cardboard , metal , wood and plastic)
- Then our output would be in the form of

```
{'cardboard': 0, 'metal': 1, 'plastic': 2, 'wood': 3}
[[0.21260688 0.29554123 0.27224338 0.21960849]
```

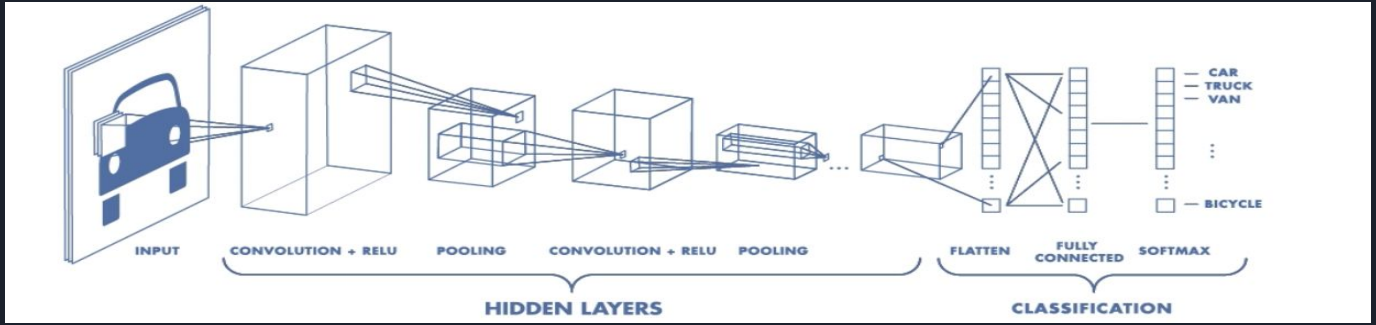


Classification Model

by Tashfeen Shahid Anwar

- We use a CNN model.
- We train it from scratch.
- We use RELU activation function alongside Convolutional Layers ,Max pooling ,Flatten and Dense layers.
- Moreover We use softmax for output.
- So far we have evaluations on two types of datasets 32x32 pixel (Val_Accuracy 40% - 50% with only 70-80 images) and 600 x 600 pixels (Val_Accuracy 25% - 30% with only 70 images (15-20 per class)

- In future we plan to improve accuracy by making a bigger dataset possibly (2000-3000 images per class). Moreover We also will make more images which have different types of dumping and label them according the percentage of trash.



Back-end

by Onen Ege Solak


```
_id: ObjectId('636cfe935b36d154a4f8cb23')
long: 33.0381593
lat: 34.732828600000005
districtId: "Limassol"
imageurl: "http://res.cloudinary.com/egesol/image/upload/v1668087432/content/test…"
type: Array
  0: "metal"
status: "suspected"
address: "Myrtou, 4552 Φασούλα Λεμεσού, Cyprus"
__v: 0
```

```
_id: ObjectId('6356e2390ac20352d1c783b6')
id: "Nicosia"
name: "Nicosia District"
password: "$2y$10$cUbr1H/iBXfXAVkPjbKyyeY6aHvDkMwZ3s1sITLk/XumrKlvPUrJS"
username: "NicoMun"
Respworkers: Array
  0: "Yannis Kalimeris"
  1: "Panos Dendias"
```

Workers

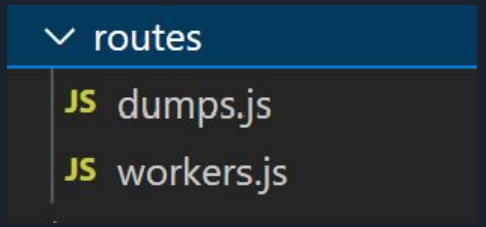
./workers

./addworker

./delworker

./login

```
router.post("/login", async (request, response) => {
  // check if email exists
  const { username, password } = request.body;
  const muni = await municipalities.findOne({ username: username });
  // if username exists
  if (muni) {
    // compare the password entered and the hashed password found
    const passwordCheck = await bcrypt.compare(password, muni.password);
    // if the passwords match
    if (passwordCheck) {
      //   create JWT token
      const token = jwt.sign(
        {
          municipalitiesId: muni.id,
          municipalitiesName: muni.username,
        },
        "RANDOM-TOKEN",
        { expiresIn: "24h" }
      );
      //   return success response
      return response.status(200).send({
        message: "Login Successful",
        username: muni.username,
        name: muni.name,
        id: muni.id,
        token: token,
      });
    } else {
      return response.status(404).send({
      });
    }
    // catch error if password does not match
  } else {
    // catch error if username does not exist
    return response.status(400).send({
    });
  }
});
```



Dumps

./getdumps

./createdump

./deldumps

./confirmdump

./suspectdump

```
function reverseGeocoding(req, res, next) {
  const { lat, long, imageurl, type } = req.body;
  console.log(req.body);
  if(lat,long,imageurl,type){
  var url =
    "https://api.mapbox.com/geocoding/v5/mapbox.places/" +
    long +
    ", " +
    lat +
    ".json?access_token=" +
    ACCESS_TOKEN;

  request({ url: url, json: true }, function (error, response, next) {
    if (error) {
      res.send("Unable to connect to Geocode API");
    } else if (response.body.features.length == 0) {
      res.send("Unable to find location. Try to" + " search another location.");
    } else {
      const address = response.body.features[0].place_name;
      const place =
        response.body.features[response.body.features.length - 2].place_name;
      const district = place.substring(0, place.indexOf(","));
      Dumps.insertMany({
        long: long,
        lat: lat,
        districtId: district,
        imageurl: imageurl,
        status: "suspected",
        address: address,
        type: type,
      });
    }
  });
  next();
}else{
  res.send("Provide all images");
}
}
```

routes

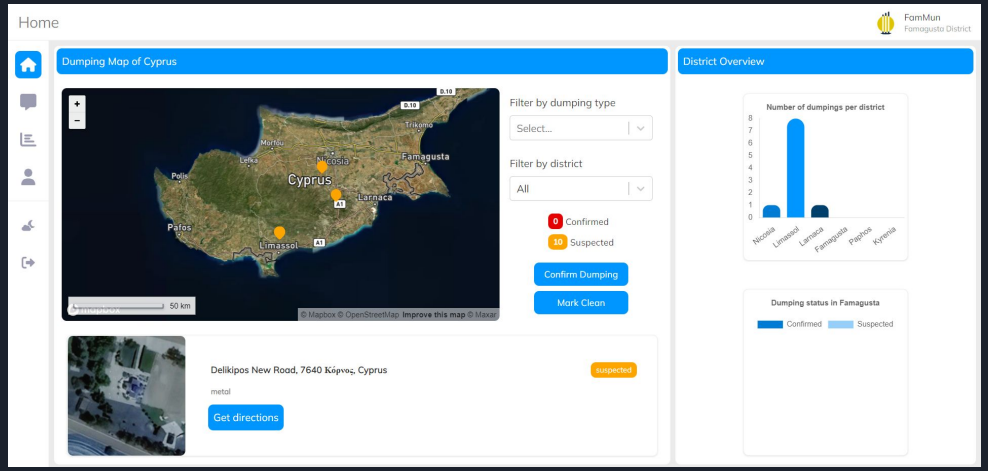
JS dumps.js

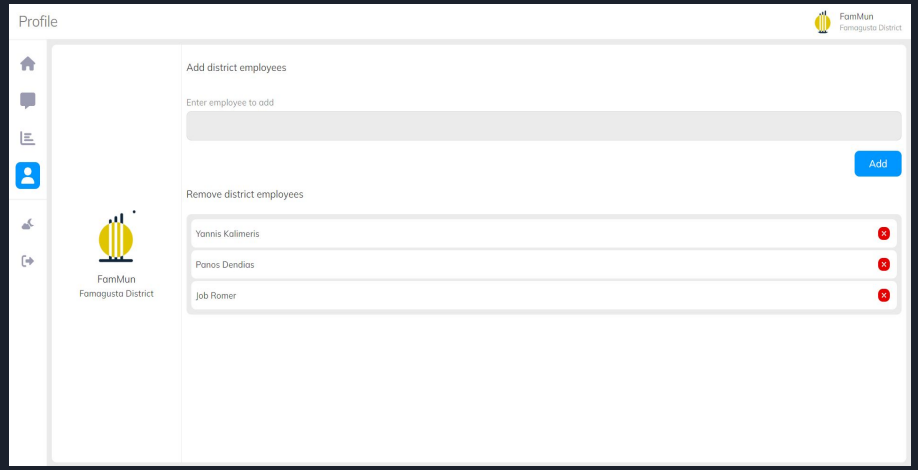
JS workers.js

Front-end

by Wishal M Sri Rangan

The web application serves as a dashboard to visualize dumping locations on an interactive map as well as display statistics on different attributes such as types of dumping and frequency of dumping per week or district. In addition to this the dashboard serves as an employee management tool to add and remove district employees.





Questions?