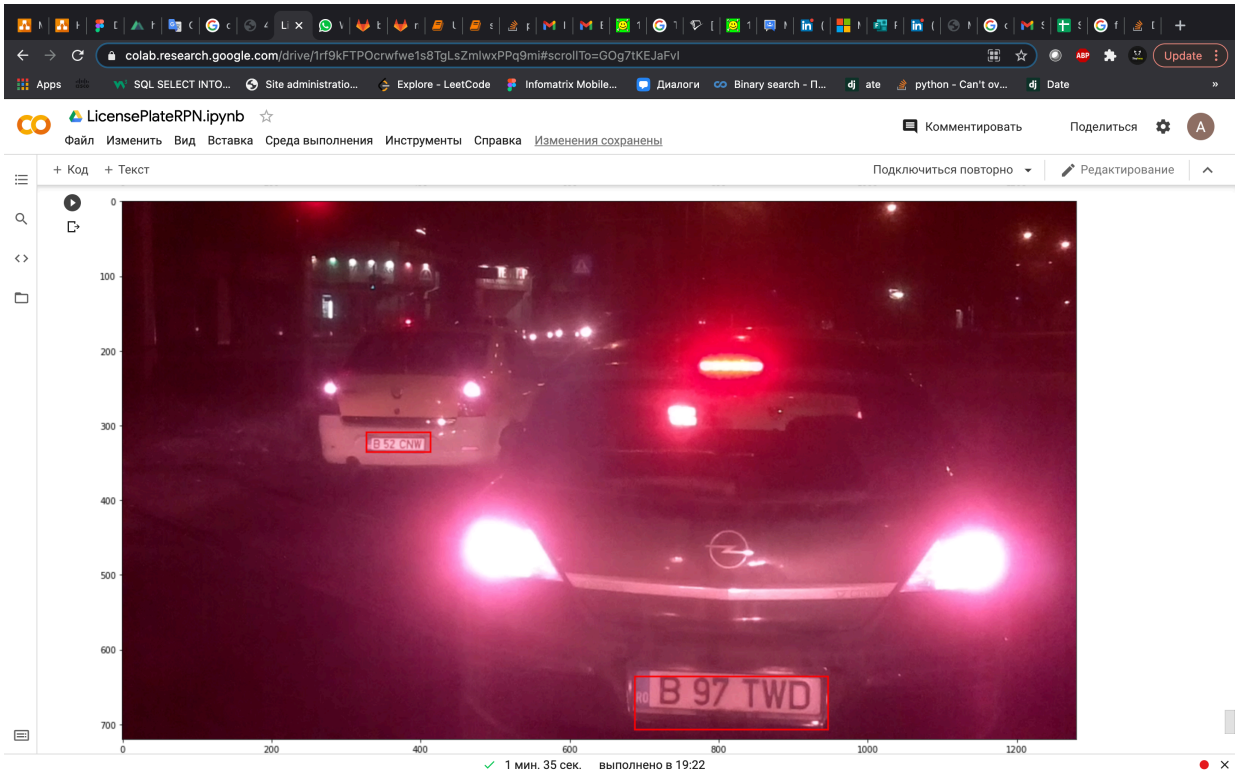
Assignment 4

Locating License Plates



Example (screenshot)

Resources

DATASET

Romanian (European Union) Dataset of License Plates

Format

A dataset of Romanian (European Union) license plates in VOC format using VOTT.

Specs

The dataset is composed of 534 images of which 80% of them are for training and the rest of 20% is for validation. Since the dataset is rather small, it is encouraged to fine-tune a preexisting model with this dataset.

The dataset was shot in both daytime and nightime.

Source

https://github.com/RobertLucian/license-plate-dataset

LINEAR SUM ASSIGNMENT

Description

The linear sum assignment problem is also known as minimum weight matching in bipartite graphs. A problem instance is described by a matrix C, where each C[i,j] is the cost of matching vertex i of the first partite set (a “worker”) and vertex j of the second set (a “job”). The goal is to find a complete assignment of workers to jobs of minimal cost.

Source

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linear_sum_assignment.html

DARKNET53

Source

https://github.com/developer0hye/PyTorch-Darknet53/blob/master/model.py

Basic Idea

** Note, here the batch size is omitted. For this assignment take the **batch_size == 1** for simplicity.*

FORWARD FUNCTION

Image (**3**x**720**x**1280**)
-
-
Model (DarkNet)
-
-
Output (**5**x**H**x**W**)
-
-
Adding prior values to the output (**5**x**H**x**W**)
-
-
Converting into the format [confidence, xmin, ymin, xmax, ymax]
-
-
Output (**5**x**H**x**W**)
-
-
Reshaping
-
-
Output (**H***x**W**x**5**)

- **H** is the height of the tensor
- **W** is the width of the tensor

LOSS FUNCTION

Loss (Output (H*Wx5), Targets (Nx4) **)**
=> Bipartite matching (H*W predictions to N targets)
=> (N) positive losses + **(H*W - N)** negative losses

- **N** is the number of grid cells

Dataset

DATASET HIERARCHY

- train
  - images
    - nightride_type3_001.mp4#t=74.jpg
    - …
  - annots
    - nightride_type3_001.mp4#t=74.xml
    - …
- valid
  - images
    - nightride_type3_001.mp4#t=715.jpg
    - …
  - annots
    - nightride_type3_001.mp4#t=715.xml
    - …

XML SAMPLE

```
<annotation verified="yes">
    <folder>Annotation</folder>
    <filename>nightride_type3_001.mp4#t=74.jpg</filename>
    <path>license-plate-detector-PascalVOC-export/Annotations/nightride_type3_001.mp4#t=74.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>1280</width>
        <height>720</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
    <name>license-plate</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
        <xmin>621.7564468882059</xmin>
        <ymin>300.983606557377</ymin>
        <xmax>736.233612658471</xmax>
        <ymax>334.37703867427626</ymax>
    </bndbox>
</object>
</annotation>
```

Best Practices

1. *Try to overfit*

1. Create Dataset from a **SINGLE SAMPLE** without any data augmentation
2. Do not apply any **randomization**
3. Try to overfit and see the result either the network **converges** or **diverges**
4. If you have found the best **hyperparameters** and the network **converges**, then move to the next step

2. *Train the network*

1. Do not change the **hyperparameters.** Add helper functions in order to save the weights and view evaluations constantly during the training process. Be aware that the training process takes about **4-12 hours**.
2. Create Dataset from all train samples with data augmentation (color jitter, noise etc).
3. If you want to use transform functions, then be aware that you have to apply transform functions to the targets as well
4. Take **batch_size == 1** for simplicity

Evaluation

1. Evaluate function to display an image, target and predicted locations - 10 points
2. Correct implementation of the forward function - 40 points
3. Correct implementation of the loss function - 40 points
4. Training the network - 10 points

Details

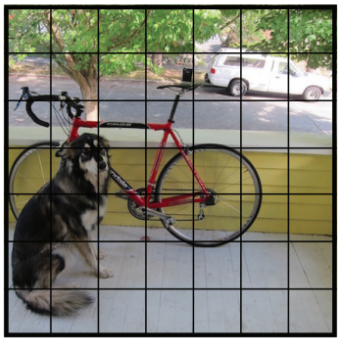
FORWARD FUNCTION

- An image, with the size of 3x720x1280, passed throw the darknet53 model. Last classifier layers are removed leaving only convolution layers.

- Here you can add some MaxPool & Convolutions layers at the end in order to decrease the **height** and **width** of the **output tensor**. In practice, **H*W == 50** showed good performance.

- *If the model predicts many anchor boxes, then the network would not be able to be trained due to imbalanced number of positive and negative anchor boxes.*

- The forward function has a tricky moment. For example a given network predicts **5** values { **confidence**, center **x**, center **y**, **height, width** } or { **c, x, y, h, w** } for each grid cell. Here, you have to add some prior values.



Prior values added in the following way

c := sigmoid (c)
x := x * grid_width + dx # **dx** - *center of the grid relative to the image*
y := y * grid_height + dy # **dy** - *center of the grid relative to the image*
h := h * grid_height
w := w * grid_width
output = [c, x - w / 2, y - h / 2, x + w / 2, y + h / 2]
 # Now, the format of the output is [c, xmin, ymin, xmax, ymax]

LOSS FUNCTION

Let's consider a specific case. The model returns 50x5 tensor, where 50 is the number of grid cells and 5 represents an anchor box, { **confidence, xmin, ymin, xmax, ymax** }. For example, a given image contains 2 license plates. So, the target tensor would have the size of 2x4, where 2 is the number of anchor boxes, and 4 represents an anchor box without confidence { **xmin, ymin, xmax, ymax** }.

Now, we have to calculate the loss function. Here, we create Cost Matrix of all combinations in order to calculate the best loss:

	a1	a2	a3	…	a50
t1	L(t1, a1)	L(t1, a2)	L(t1, a3)		L(t1, a50)
t2	L(t2, a1)	L(t2, a2)	L(t2, a3)		L(t2, a50)

```
L(t[i], a[j]) = confidence_loss (a[j][0]) + location_loss (t[i][:], a[j][1:])

confidence_loss (x) = −log(x + 0.0000000001)

location_loss (a, b) = MSE (a, b)
```

We have to assign our targets to two anchor boxes in the way to get the minimum cost. This problem is named **bipartite matching**. It can be solved using the Hungarian algorithm. But, for simplicity use already made function from the given source: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linear_sum_assignment.html.

Now, we marked 2 anchor boxes from 50 as positive ones, but you have to calculate loss function for negative ones, too. Here, you **do not need** to calculate the location loss for negative ones. A simple loss function for negative anchor boxes is given below

```
loss (a) = −log(1 − a[0] + 0.0000000001)
```

Let's summarize

1. We select 2 best anchor boxes (because we have 2 targets) from 50 predicted anchor boxes
2. We calculate total loss

```
total_loss = positive_loss (2 best anchor boxes, 2 targets) + negative_loss (48 anchor boxes, no targets)
```

Note, loss function is imbalanced !!!!!

1. First case, you have 2 positive anchor boxes and 48 negative ones.
2. Second case, you have 2 different scaled loss functions: **confidence_loss** and **location_loss**. In practice, **confidence_loss** varies from 0 till 24, but **location_loss** varies from 0 till 10000 in average.

You have to balance them gently in order to have a smooth gradient descent and a total convergence !!!

The loss function with weights would like

```
positive_loss = W_conf ∗ confidence_loss + W_loc ∗ location_loss
negative_loss = one_minus_confidence_loss = −log(1 − a[0] + 0.0000000001)
total_loss = W_pos ∗ positive_loss + W_neg ∗ negative_loss
```

Submission policy

You have to submit 3 files. Put 2 files, **source.ipynb** and **source.html**, in a single folder named with your **student id**, for example 180102030. The folder should be compressed in ZIP format. Send 3rd file, **model.pt**, in other ways because the size can be large.

180102030/

1. **source.ipynb**

   Please write your name and surname on the top in addition

2. **source.html** *(you can miss this file, but I suggest you to upload it as well. If source code would have problems, then this file guarantees your work done)*

   This is HTML version of your source code. You can export the HTML version of your work in Jupyter Notebook.

3. **model.pt**

   Do not forget to save the state of you WEIGHTS!!!! You can read more about it from: https://pytorch.org/tutorials/beginner/saving_loading_models.html.

   If the moodle do not allow you upload big files, then upload it into your google drive and add the link into your work, or share it in other ways.