License Plate Detection and Modification

Group 7: Xiule Fan Shiva Tej Soma Xinyi Yao

Problem Formation

Practical Problem:

Recently, with the spread of car videos across the internet or television broadcasts, there are concerns about the privacy violation toward drivers since we can clearly see the vehicle license plate information through videos.

Importance of addressing above problem:

Addressing this problem can effectively protects vehicle license plate information and further protects the privacy of drivers.

Reason of Interest:

Our interest in solving this problem stems from the desire to promote privacy protection and facilitate video content creation without compromising the security and privacy of individuals.

Project Goal and Milestones

Project Goal:

We aim to develop a system that can detect and modify the license plates of cars in videos so that the license plate information will not be leaked.

Task breakdown:

- Milestone 1
  - License Plate Detection From Images
  - License Plate Detection From Videos
- Milestone 2
  - License Plate Modification In Images(inpainting/diffusion)

Datasets Used

YOLOv3:

- Kaggle dataset
- RoboFlow dataset
- 678 training images, 70 validation images, 35 test images

YOLOv5:

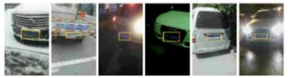
- Kaggle dataset of 473 images
- CCPD (Chinese City Parking Dataset)
    - This dataset consists of 250k+ images, including all kinds of augmentations.

CCPD (Chinese City Parking Dataset)

Edit

Introduced by Xu et al. in Towards End-to-End License Plate Detection and Recognition: A Large Dataset and Baseline

The **Chinese City Parking Dataset** (CCPD) is a dataset for license plate detection and recognition. It contains over 250k unique car images, with license plate location annotations.

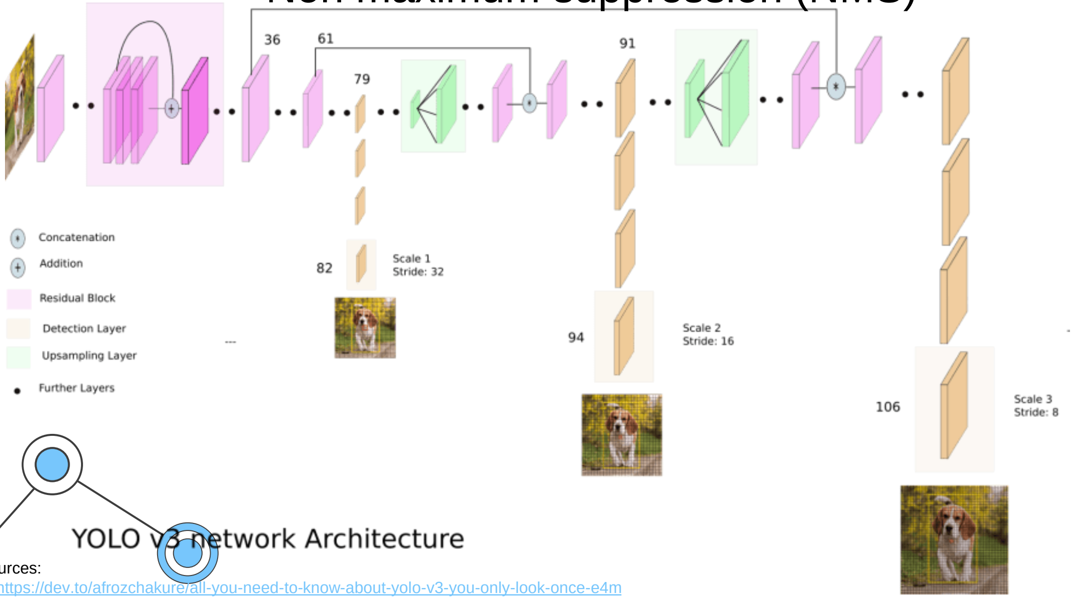


Source: Xu et al.

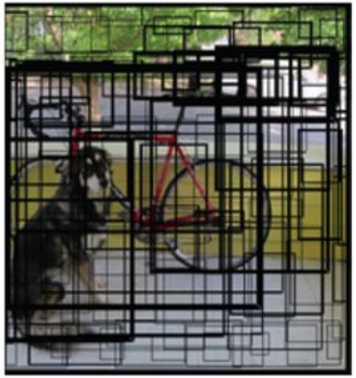
Source: Towards End-to-End License Plate Detection and Recognition: A Large Dataset and Baseline

License Plate Detection from Images

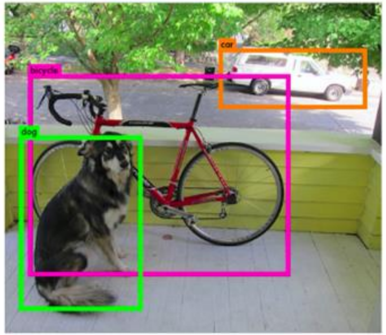
- YOLOv3 - Model
  - Architecture
  - Non maximum suppression (NMS)



YOLO v3 network Architecture



Multiple Bounding Boxes



Final Bounding Boxes

Sources:
1. https://dev.to/afrozchakure/all-you-need-to-know-about-yolo-v3-you-only-look-once-e4m
2. https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

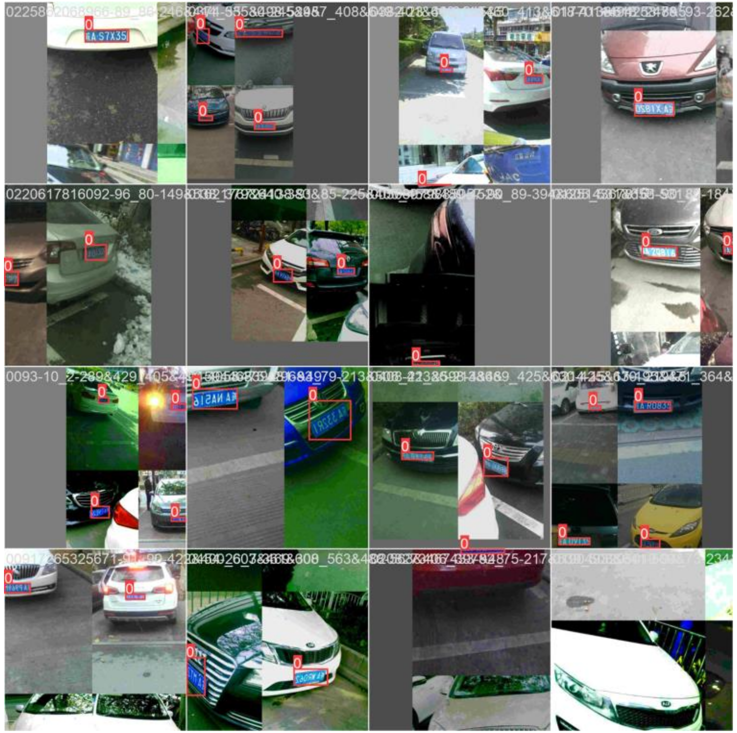
License Plate Detection from Images

- YOLOv3 - Result on images
  - Results: Average Precision (AP) = 0.7525

License Plate Detection from Images

- YOLOv5 - Result on images

On CCPD2019 Dataset



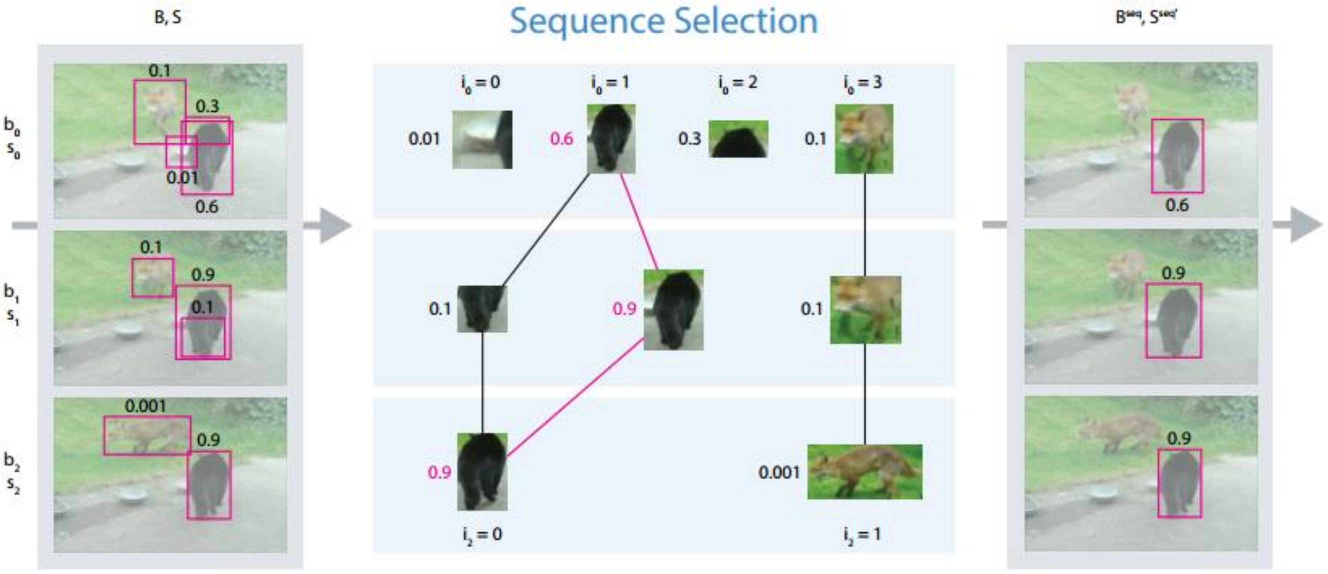
On Kaggle Dataset

License Plate Detection from Images

- YOLOv3 - Result on videos
  - False positive detections
  - Missing detections at some frames
- YOLOv5 - less consistent plate detections

License Plate Detection from Videos

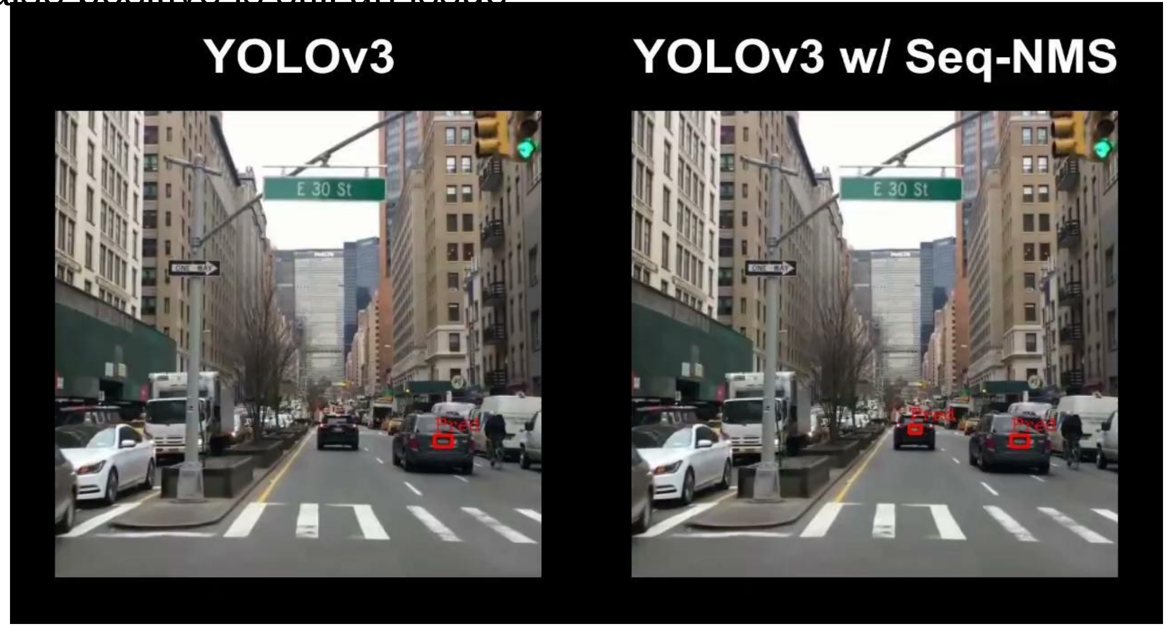
- Seq-NMS
  - An extension of NMS for image sequences



Sources:
1. W. Han et al., “Seq-NMS for Video Object Detection,” arXiv:1602.08465 [cs.CV], 2016.

License Plate Detection from Videos

- YOLOv3 with Seq-NMS
  - More consistent prediction throughout the sequence
  - False positive is still an issue



License Plate Detection and Modification

Click the image to open the Video

MILESTONE 2

Blurring license plates may be an acceptable solution to protect privacy, but it can cause distraction and shift the focus from the main subject of the image. To address this issue, we conducted further research to find alternative solutions that would maintain the privacy of the individuals while also improving the overall aesthetics of the image.

We Tried Inpainting and Diffusion

Inpainting

Image Inpainting is a task of reconstructing missing regions in an image. It is an important problem in computer vision and an essential functionality in many imaging and graphics applications, e.g. object removal, image restoration, manipulation, re-targeting, compositing, and image-based rendering.
E.g: Healing tools in photoshop or snapseed

Inpainting example, google photos

GAN Architectures for Inpainting

Used both CCPD and Kaggle datasets on below architecture variations (trained for 100 epochs each)

- Custom Generator and discriminator
- Pretrained U-Net with resnet encoder, custom discriminator
- Freezing few layers for pre-trained U-Net.

Custom GAN Architecture

Generator

```
Input Image (3 channels)
↓
Conv2d(3, 64) → ReLU
↓
Conv2d(64, 128) → BatchNorm2d → ReLU
↓
Conv2d(128, 256) → BatchNorm2d → ReLU
↓
ConvTranspose2d(256, 128) → BatchNorm2d → ReLU
↓
ConvTranspose2d(128, 64) → BatchNorm2d → ReLU
↓
ConvTranspose2d(64, 3) → Tanh
↓
Output Image (3 channels)
```

Discriminator

```
Input Image (3 channels)
↓
Conv2d(3, 64) → LeakyReLU(0.2)
↓
Conv2d(64, 128) → BatchNorm2d → LeakyReLU(0.2)
↓
Conv2d(128, 256) → BatchNorm2d → LeakyReLU(0.2)
↓
Conv2d(256, 1) → AdaptiveAvgPool2d → Sigmoid
↓
Output (1 channel)
```

Generator

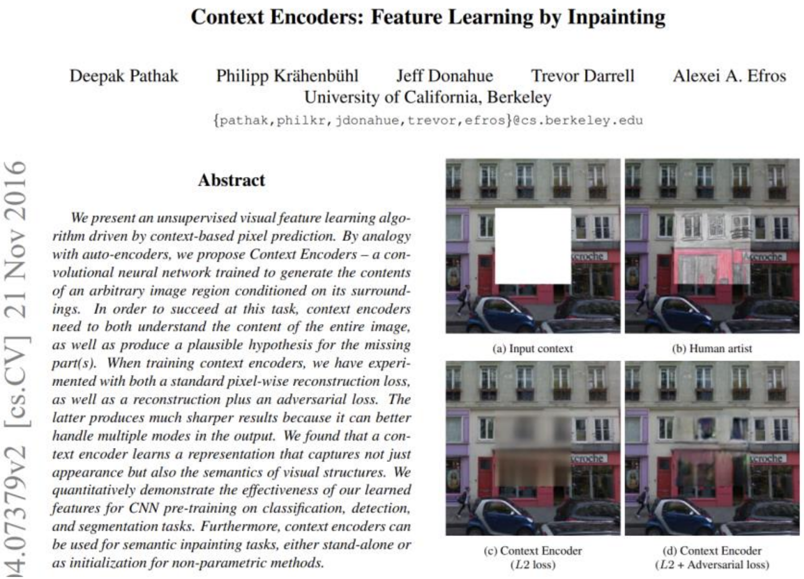
```
Input Image (3 channels)
↓
Conv2d(3, 64) → ReLU
↓
Conv2d(64, 128) → BatchNorm2d → ReLU
↓
Conv2d(128, 256) → BatchNorm2d → ReLU
↓
ConvTranspose2d(256, 128) → BatchNorm2d → ReLU
↓
ConvTranspose2d(128, 64) → BatchNorm2d → ReLU
↓
ConvTranspose2d(64, 3) → Tanh
↓
Output Image (3 channels)
```

Discriminator

```
Input Image (3 channels)
↓
Conv2d(3, 64) → LeakyReLU(0.2)
↓
Conv2d(64, 128) → BatchNorm2d → LeakyReLU(0.2)
↓
Conv2d(128, 256) → BatchNorm2d → LeakyReLU(0.2)
↓
Conv2d(256, 1) → AdaptiveAvgPool2d → Sigmoid
↓
Output (1 channel)
```

Observations

- All models resulted in Blurry outputs
- According to the Research paper from 2016 (University of California) L2 loss prefers blurred images therefore it is suggested to divide losses into reconstruction loss and adversarial loss



Context Encoders: Feature Learning by Inpainting

Deepak Pathak Philipp Krähenbühl Jeff Donahue Trevor Darrell Alexei A. Efros
University of California, Berkeley
{pathak,philkr,jdonahue,trevor,efros}@cs.berkeley.edu

4.07379v2 [cs.CV] 21 Nov 2016

Abstract

We present an unsupervised visual feature learning algorithm driven by context-based pixel prediction. By analogy with auto-encoders, we propose Context Encoders – a convolutional neural network trained to generate the contents of an arbitrary image region conditioned on its surroundings. In order to succeed at this task, context encoders need to both understand the content of the entire image, as well as produce a plausible hypothesis for the missing part(s). When training context encoders, we have experimented with both a standard pixel-wise reconstruction loss, as well as a reconstruction plus an adversarial loss. The latter produces much sharper results because it can better handle multiple modes in the output. We found that a context encoder learns a representation that captures not just appearance but also the semantics of visual structures. We quantitatively demonstrate the effectiveness of our learned features for CNN pre-training on classification, detection, and segmentation tasks. Furthermore, context encoders can be used for semantic inpainting tasks, either stand-alone or as initialization for non-parametric methods.

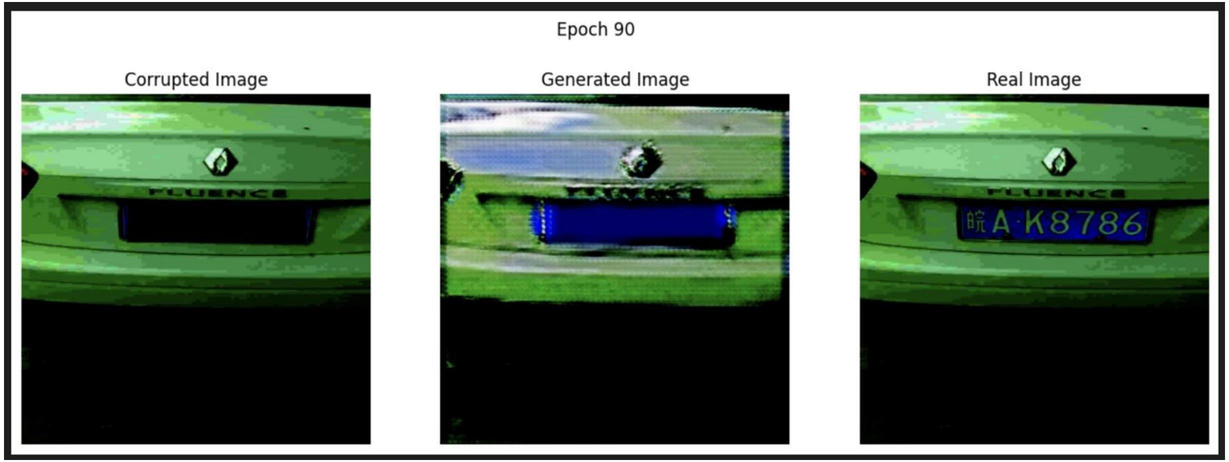
(a) Input context (b) Human artist

(c) Context Encoder (L2 loss) (d) Context Encoder (L2 + Adversarial loss)

1604.07379.pdf (arxiv.org)

Observations

- All of the Algorithms did an ok job inpainting, but at the cost of overall decrease in quality
- Most often the they are just filled with dominant color.



Custom GAN output

Observations

- When used pre-Trained U-Net with ResNet34 as encoder

Which one is generated?

Observations

- When used pre-Trained U-Net with ResNet34 as encoder

Real

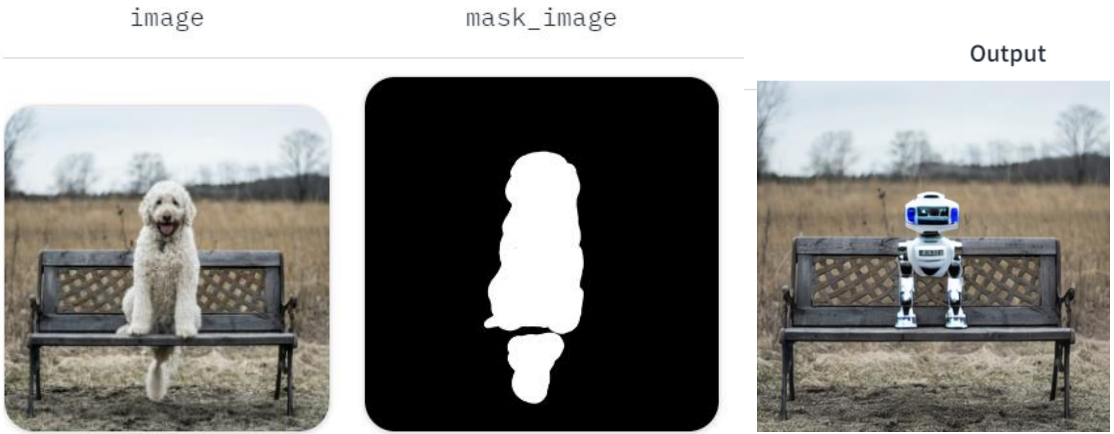
Generated

Generated image colors looks realistic

Diffusion

Diffusion:

- Removes data by slowly adding gaussian blur
- Train a neural network to reverse the corruption process.
- It's a fascinating approach that has proved quite effective in Image Generation tasks
- E.g: DALLE2, Midjourney, Stable Diffusion



Diffusion inpainting example

Idea

Real Image

Idea

Real Image

Corrupted Image

Idea

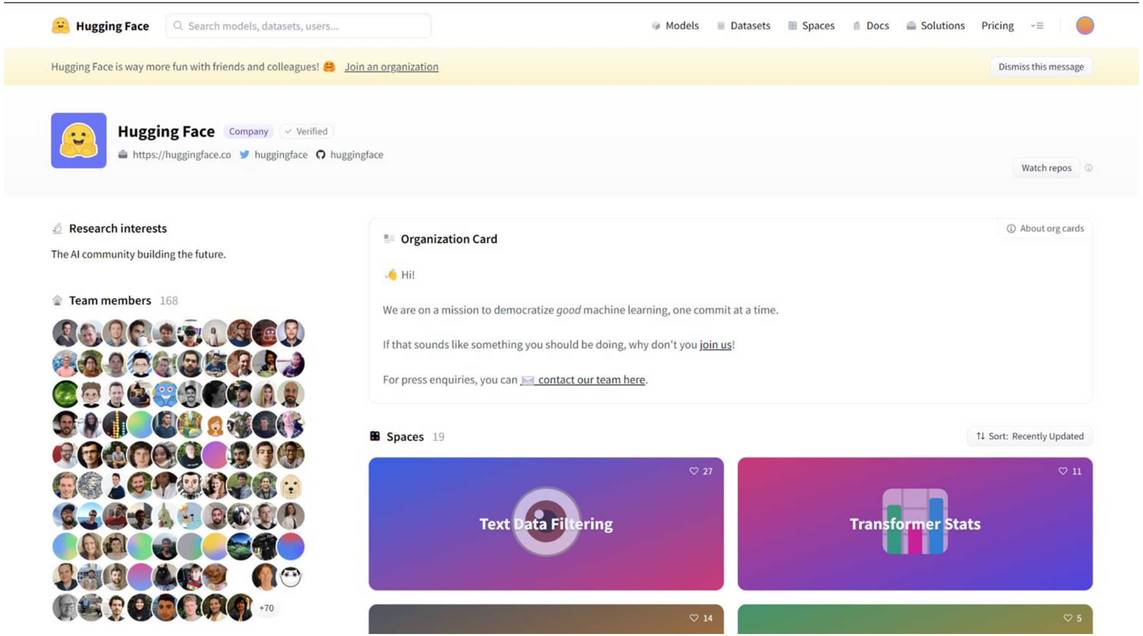
Real Image

Corrupted Image

Generated Image

Tool used: Hugging face (Diffuser)

Consists of open source AI models and Tools to build them | diffuser module (2022)



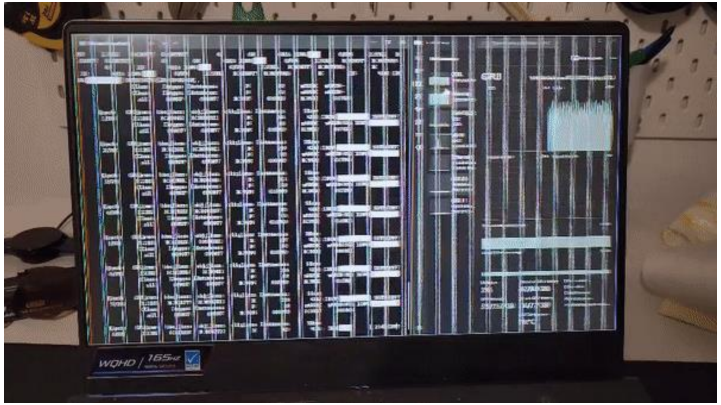
Results trained on license plates alone

Text is not legible

Discussion

What we have learned from this project:

- Data augmentation
- Thresholds for different parameters
- Anchor box size
- Consumer Hardware Limitations
- Reading Research Paper
- Trick: Leverage parallel computing for cpu bound tasks (data preparation/augmentation)



PC crash when training

Thanks!

Many more test videos and Notebook file in form of tutorial are available.
Feel free to ask

😄

License Plate Detection from Videos

- YOLOv3 with Seq-NMS