Automatic Meter Reading Using YOLO

Abstract:

Automatic instrument reading has become an increasingly essential problem for intelligent sensors in smart cities. Thanks to the rapid development of artificial intelligence and fifth-generation mobile network technologies. In this paper, we propose an end-to-end AMR strategy that focuses on unconstrained settings also we have created our own dataset of cropped images of counter. Existing image-based Automatic Meter Reading (AMR) techniques have been tested on images captured in well-controlled situations. Existing approaches become unresponsive when dealing with fuzzy, hazy, or blurry meter images. We have used two main components to our metre reading system: i) You Only Look Once (YOLO) is used to identify counter region, ii) Digit Detection using YOLO. Using the method provided here, it is possible to reliably detect and recognise different metre kinds. Furthermore, because real-world applications accept a few reading errors, we show that our AMR system achieves remarkable recognition rates of 99% even while rejecting readings with lower confidence values.

I. Introduction:

Automatic metre reading (AMR) is the technology of automatically detecting consumption, diagnostic, and status data from water metres or energy metering devices and storing it in a central database for billing, troubleshooting, and further analysis. This technique primarily saves utility providers money by eliminating the cost of making frequent trips to each physical location to read metres. Another advantage is that billing can be based on near real-time consumption rather than on estimates based on past or predicted consumption. This real-time data, combined with analysis, can help utility companies and customers manage the use and production of electric energy, gas, and water more effectively. The idea behind image-based AMR, which is a special scenario of scene text detection and identification, is that the inspection may be carried out automatically, decreasing human errors and conserving manpower.

Handheld, mobile, and network technologies based on telephony platforms (wired and wireless), Radio Frequency (RF), or powerline transmission are examples of AMR technologies.

Despite the fact that image-based AMR has gotten a lot of attention in recent years, most studies in the literature are still constrained in numerous ways. The studies were carried out on either proprietary datasets or datasets containing photos taken in well-controlled conditions. In contrast, in related study areas like automatic licence plate recognition, attention has switched in recent years to unconstrained scenarios (with problematic elements like blur, various lighting conditions, size variations, in-plane and out-of-plane rotations, occlusions, and so on), which significantly advance the art's status. Furthermore, many works focus on a single stage of the AMR pipeline, making it impossible to fully evaluate the methods provided from start to finish

(for example, the results obtained by a recognition model may vary significantly depending on how accurate the counter region is recognized).

In this study, we propose a three-stage AMR strategy. To get the ultimate result, we first identify the counter region, then trim the discovered region and create a new dataset with cropped photos. Finally, we use yolov4 to detect the individual digits.

It's an object recognition system that can identify numerous items in a single frame in real time. YOLO identifies things more accurately and quickly than previous systems. It can identify up to 9000 classes, including previously undiscovered ones. It's an object recognition system that can identify numerous items in a single frame in real time. YOLO identifies things more accurately and quickly than previous systems.

Based on what we've talked about so far, our work is summarized as:

- CNNs are used for both counter detection and recognition in a two-stage AMR method.
- Manually annotated 7,877 images of metre for counter detection.
- Custom function is utilised for detection of counter and cropping of the counter region, thus making new dataset of cropped counter.
- New dataset formed of cropped counter again annotated for each digit, trained thus forming custom Optical Character Reader (OCR) for digits.

II. Related Work:

Over the last decade, various methods for automatic metre reading using metre images have been proposed. Several strategies were utilised in the previously proposed procedures, such as augmentation, contouring, segmentation, and so on, but the results were not as effective as those produced using deep learning architects.

Image-based AMR is a type of scene text detection and recognition application. However, there are some fundamental differences for the overall goal of detecting and recognizing scene text that should be highlighted: (i) In AMR, each image has only one region of interest (the counter region), and all digits are contained within it; (ii) AMR recognition networks must learn 10 classes (digits 0 to 9), whereas general scene text recognition networks must learn 36 character classes (26 letters and 10 digits; the number of classes can be even higher depending on the language); and (iii) AMR presents an unusual challenge in Optical Character Recognition (OCR). Even when sophisticated approaches for digit/counter recognition are used, rotating digits are typically a primary source of errors in such metres.

This last argument underscores the importance of the proposed dataset, as robust reading recognition models trained solely on images from general datasets (e.g., ICDAR 2013) are likely to fail in these circumstances.

Here's a breakdown of the linked works we looked at.

The task of localization and recognizing Racing Bib Images (RBI) dealt by **Wong et al. [1],** achieve the accuracy of the Mean Average Precision (mAP) is 85.15 % but in the unconstrained setting the result are not promising.

A deep learning model that was employed by **Son et al. [2]** provides an AMR system for efficient metre reading. This system has an end-to-end accuracy of 85.71 % for the counter region and 60.90 % for the ID region.

In the year 2019, **R. Laroca et al. [3]** introduced a two-stage AMR approach in which CNNs were employed for both counter detection and recognition, with an accuracy of 94.13%, with a reasonable FPS only on high-end graphics cards. **R. Laroca et al. [4]** continued their work and in 2021 modelled another AMR system and introduced a new technique called corner detection and counter classification, increasing accuracy in the UFPR-AMR and Copel-AMR datasets of the metre images to 94.75% and 96.98%, respectively.

Li et al. [5] suggested a light-weight CNN for counter recognition that splices a certain number of 1 1 and 3 3 kernels to minimise network parameters with little loss in recognition rate, taking into account the necessity of constructing highly efficient algorithms in the AMR environment. The results they reported are impressive considering the accuracy/speed trade-off achieved; however, their experiments were conducted exclusively on a private dataset with well-controlled images that are very similar to each other (i.e., the images were captured by a camera installed in the metre box and manually pre-processed by the authors; thus, they have no blur, scale variations, shadows, occlusions, significant rotations, or other challenging factors).

For counter recognition, **Marques et al.** [6] fine-tuned the Faster R-CNN and RetinaNet object detectors. Although the authors claimed mean Average Precision (mAP) rates of over 90% for both detectors, they only used a small subset of counter images from a private dataset in their experiments, and the hardware used (i.e., the GPU) was not specified, making it difficult to compare their methodology to previous works in terms of efficiency and recognition rate.

Counter detection has been extensively studied using object detectors. For example, **Koscevi c &'Subasi** [7] used Faster R-CNN to recognise counters and serial numbers on images of residential metres, whereas **Tsai et al.** [8] used a fine-tuned Single Shot MultiBox Detector (SSD) to detect counters in electricity metres. Only private datasets were used to assess the detectors in both trials.

III. About Dataset:

The AMR dataset contains over 8,000 metre images gathered in the field by Energy Company staff. The metre images are of various Indian households, including districts, villages, and settlements. As a result, AMR is made up of photos taken in unrestricted situations, which generally contain blur (due to camera movement), dirt, scale changes, in-plane and out-of-plane rotations, reflections, shadows, and occlusions. Because of obstructions or broken meters, the metre reading can't be done in 2,000 photos, or 25% of the dataset. The dataset contained several hazy pictures, which we eliminated by hand to get the total number of images up to 7877. Then, in order to facilitate quick access while we were identifying the images, we changed the names of the images from 1 to 7,877. The images have resolutions of either 2992 x 4000 or 3072 x 4096, respectively. We went through and manually labelled the metre reading, the position of each of the four corners of the counter, and made a bounding box (x, y, w, h) for the counter region in our dataset. We manually labelled 5,537 digits since the AMR dataset comprises 7,877 photos of legible or operational metres, and each metre reading consists of 7 digits. We noticed that the digit '0' appears far more frequently than the others, which is to be expected given that a brand-new metre starts with 00000 and the leftmost digit

positions take longer to increase. As a result, the frequency of numbers decreases with increasing numbers (i.e., 0 has the highest frequency and 9 has the lowest).

Figure 1. Sample AMR images

IV. Deep Learning:

Deep has evolved gradually over the last decade. According to one relevant definition, deep learning is defined as a "neural network with more than two layers.". We believe that before demonstrating the astounding outcomes witnessed in recent years, neural networks had to evolve structurally from prior network designs (along with a lot more processing power). The following are some of the aspects of neural network evolution:

- Network has more neurons than prior networks.
- In NNs, there are more complicated techniques to connect layers/neurons.
- Massive increase in processing power available for training
- Automatic extraction of features

Deep learning will be defined as neural networks with a large number of parameters and layers in one of four core network architectures:

- Pre-trained unsupervised networks
- Neural networks with convolutions
- Neural networks with recurrent connections
- Neural networks that recurse

A significant advantage of deep learning over typical machine learning methods is automatic feature extraction. The network's method of selecting which properties of a dataset can be used as indicators to properly classify that data is referred to as feature extraction. Machine learning practitioners have traditionally spent months, years, and even decades manually constructing exhaustive feature sets for data classification. State-of-the-art machine learning algorithms have absorbed decades of human effort as they amassed significant attributes by which to classify input at the time of deep learning's Big Bang in 2006. For nearly a decade, deep learning has outperformed traditional algorithms in terms of accuracy in every data type with

minimal human effort and adjusting. These deep networks can assist data science teams by saving time and effort for more important tasks.

Convolution Neural Networks (CNN or ConvNets) have been utilised in this work because of its ability to create a two-dimensional internal representation of an image. Yolov4 comprises of 53 convolution layers.

V. Convolution Layer

CNN is a form of deep learning model for processing data with a grid pattern, such as pictures, that is inspired by the structure of animal visual cortex and meant to learn spatial hierarchies of features, from low-to high-level patterns, automatically and adaptively. Convolutional neural networks use principles from linear algebra, notably matrix multiplication, to discover patterns inside an image, making them more scalable for image classification and object recognition tasks.

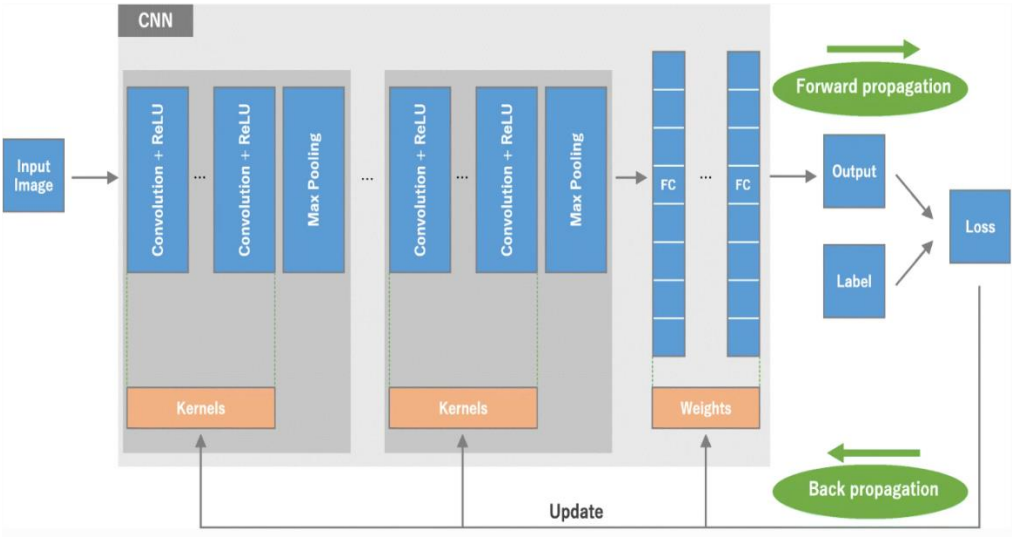

Figure 2. CNN architecture

Convolution layers, pooling layers (e.g., max pooling), and fully connected (FC) layers are the building blocks that make up a CNN. The performance of a model with certain kernels and weights is calculated using a loss function and forward propagation on a training dataset, and learnable parameters, such as kernels and weights, are updated using the gradient descent optimization process. ReLU stands for rectified linear unit.

VI. Proposed Methodology

The proposed approach is divided into three main part i.) Counter Detection ii.) Localisation of counter region and cropping iii.) Digit Detection and Recognition. Given an input image, the YOLOv4 model is utilised to determine the counter region. After that, we crop the counter using the detector's coordinates. Finally, counters are delivered to our custom-built OCR recognition network.

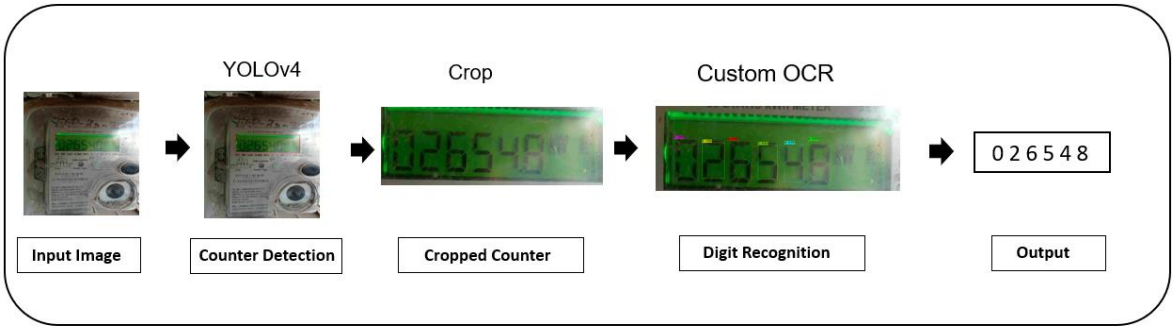

Figure 3. Sample Approach

6.1. Counter detection

Locating the counter directly in the input image in uncontrolled circumstances is difficult for three reasons: (i) due to complex backgrounds caused by dirt, reflections, and other factors, some text blocks (e.g., metre specifications and serial numbers) are very similar to the counter region in certain metre models; (ii) the counter region may occupy a very small portion of the input; and (iii) some text blocks (e.g., metre specifications and serial numbers) are very similar to the counter region in certain metre models. YOLOv4 is used in this step.

YOLOv4 consists of:

• **CSPDarknet53** as the backbone

• **SPP, PAN** for the neck

• **YOLOv3** for the head

CSPDarknet53:

CSPDarknet is based on Cross-Stage Partial Network (CSPNet), which helps to achieve a richer gradient combination while reducing the amount of computation. The main purpose of CSPNet is to strengthen the learning ability of a CNN, as the accuracy of previously existing CNNs degraded after light weighting. It also aids in the removal of computational bottlenecks and significantly reduces memory costs.

SPP, PAN:

The SPP is changed in YOLO to keep the output spatial dimension. A sliding kernel of size 11, 55, 99, or 1313 is given a maximum pool. The spatial dimension is intact. The output is created by concatenating the feature maps from different kernel sizes.

PANet is found in the YOLOv4 model's neck, and it is primarily used to improve the process of instance segmentation by conserving spatial information.

YOLOv3:

YOLOv3 is based on a Darknet variation with a 53-layer network trained on Imagenet.

YOLOv4 uses:

- Bag of Freebies (BoF) for backbone: CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing
- Bag of Specials (BoS) for backbone: Mish activation, Cross-stage partial connections (CSP), Multi-input weighted residual connections (MiWRC)
- Bag of Freebies (BoF) for detectors: CIoU-loss,CmBN, DropBlock regularization. Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyper-parameters Random training shapes
- Bag of Specials (BoS) for detector: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

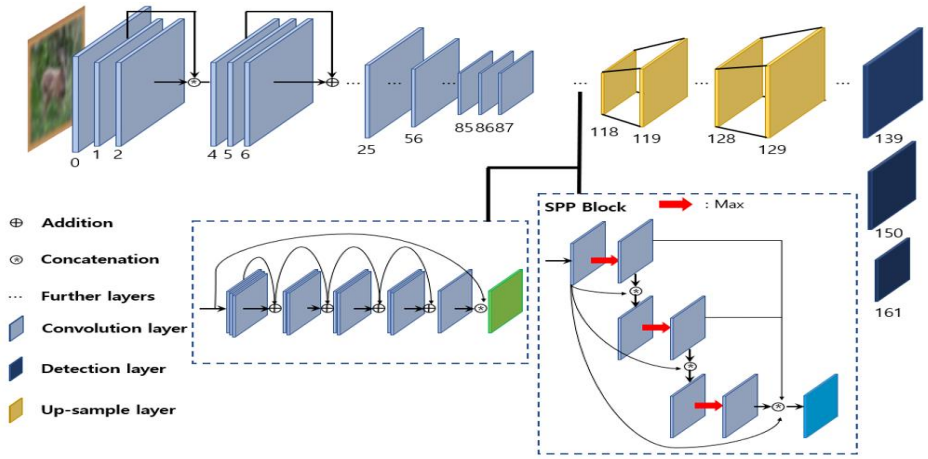

Figure 4 Yolov4 Architecture

As a result, the yolov4 model is used to detect the metre reading region in an image. To prepare the training and testing datasets, the manually labelled set of 7877 images which is divided into an 80:20 ratio. The training set has 6302 images, while the testing set has 1575 images that is ready for training on the yolov4 model. The training takes approx. 10 hours, and state of the art accuracy of 99.74% has been achieved for all types of images, including the problematic ones that are blurry and unclear.

Figure 5 Counter Detection

Table 1: Yolov4 model to predict the bounding boxes in the images

# layer	filters	size	input	output
0 conv	32	3 x 3/ 1	416 x 416 x 3	-> 416 x 416 x 32 0.299 BF
1 conv	64	3 x 3/ 2	416 x 416 x 32	-> 208 x 208 x 64 1.595 BF
2 conv	64	1 x 1/ 1	208 x 208 x 64	-> 208 x 208 x 64 0.354 BF
3 route	1			-> 208 x 208 x 64
4 conv	64	1 x 1/ 1	208 x 208 x 64	-> 208 x 208 x 64 0.354 BF
5 conv	32	1 x 1/ 1	208 x 208 x 64	-> 208 x 208 x 32 0.177 BF
6 conv	64	3 x 3/ 1	208 x 208 x 32	-> 208 x 208 x 64 1.595 BF
7 Shortcut Layer: 4, wt = 0, wn = 0, outputs: 208 x 208 x 64 0.003 BF				
8 conv	64	1 x 1/ 1	208 x 208 x 64	-> 208 x 208 x 64 0.354 BF
9 route	8 2			-> 208 x 208 x 128
10 conv	64	1 x 1/ 1	208 x 208 x 128	-> 208 x 208 x 64 0.709 BF
11 conv	128	3 x 3/ 2	208 x 208 x 64	-> 104 x 104 x 128 1.595 BF
12 conv	64	1 x 1/ 1	104 x 104 x 128	-> 104 x 104 x 64 0.177 BF
13 route	11			-> 104 x 104 x 128
14 conv	64	1 x 1/ 1	104 x 104 x 128	-> 104 x 104 x 64 0.177 BF
15 conv	64	1 x 1/ 1	104 x 104 x 64	-> 104 x 104 x 64 0.089 BF
16 conv	64	3 x 3/ 1	104 x 104 x 64	-> 104 x 104 x 64 0.797 BF
17 Shortcut Layer: 14, wt = 0, wn = 0, outputs: 104 x 104 x 64 0.001 BF				
18 conv	64	1 x 1/ 1	104 x 104 x 64	-> 104 x 104 x 64 0.089 BF
19 conv	64	3 x 3/ 1	104 x 104 x 64	-> 104 x 104 x 64 0.797 BF
20 Shortcut Layer: 17, wt = 0, wn = 0, outputs: 104 x 104 x 64 0.001 BF				
21 conv	64	1 x 1/ 1	104 x 104 x 64	-> 104 x 104 x 64 0.089 BF
22 route	21 12			-> 104 x 104 x 128
23 conv	128	1 x 1/ 1	104 x 104 x 128	-> 104 x 104 x 128 0.354 BF
24 conv	256	3 x 3/ 2	104 x 104 x 128	-> 52 x 52 x 256 1.595 BF
25 conv	128	1 x 1/ 1	52 x 52 x 256	-> 52 x 52 x 128 0.177 BF
26 route	24			-> 52 x 52 x 256
27 conv	128	1 x 1/ 1	52 x 52 x 256	-> 52 x 52 x 128 0.177 BF
28 conv	128	1 x 1/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.089 BF
29 conv	128	3 x 3/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.797 BF
30 Shortcut Layer: 27, wt = 0, wn = 0, outputs: 52 x 52 x 128 0.000 BF				
31 conv	128	1 x 1/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.089 BF
32 conv	128	3 x 3/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.797 BF
33 Shortcut Layer: 30, wt = 0, wn = 0, outputs: 52 x 52 x 128 0.000 BF				
34 conv	128	1 x 1/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.089 BF
35 conv	128	3 x 3/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.797 BF
36 Shortcut Layer: 33, wt = 0, wn = 0, outputs: 52 x 52 x 128 0.000 BF				
37 conv	128	1 x 1/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.089 BF
38 conv	128	3 x 3/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.797 BF
39 Shortcut Layer: 36, wt = 0, wn = 0, outputs: 52 x 52 x 128 0.000 BF				
40 conv	128	1 x 1/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.089 BF
41 conv	128	3 x 3/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.797 BF
42 Shortcut Layer: 39, wt = 0, wn = 0, outputs: 52 x 52 x 128 0.000 BF				
43 conv	128	1 x 1/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.089 BF
44 conv	128	3 x 3/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.797 BF

45 Shortcut Layer: 42, wt = 0, wn = 0, outputs: 52 x 52 x 128 0.000 BF

CUDA-version: 11010 (11020), cuDNN: 7.6.5, CUDNN_HALF=1, GPU count: 1
OpenCV version: 3.2.0
0 : compute_capability = 370, cudnn_half = 0, GPU: Tesla K80

Table 2: Results

[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
Total BFLOPS 59.563
avg_outputs = 489778
Accuracy: 98%

6.2. Localisation of Meter region and cropping

We utilise the weights to detect the metre reading region for all the images and acquire the coordinates of the corner points of the detected region once we have the yolov4 weights that yield good accuracy after training. Then, using these coordinates, we crop the metre reading area region from the rest of the image, which is useless for our needs. As a result, another set of cropped images is created in this stage, which will be used in subsequent steps to train a new digit detection and identification model.

Figure 6 Sample Cropped Images

6.3. Digit Detection and recognition

For digit detection, the cropped images of metre readings produced in the previous stage are now utilised. We created another set of labelling datasets for digit detection, consisting of 1000 cropped photos with digit labelling from 0–9. The digit recognition labelled dataset was trained using the yolov4 model, which achieved an accuracy of 87.4%. The training included a wide range of cropped photos, some of which were so small that they were almost impossible to read.

Table 3: Yolov4 model for digit detection from the cropped images

# layer	filters	size	input	output
0 conv	32	3 x 3/ 1	416 x 416 x 3	-> 416 x 416 x 32 0.299 BF
1 conv	64	3 x 3/ 2	416 x 416 x 32	-> 208 x 208 x 64 1.595 BF
2 conv	64	1 x 1/ 1	208 x 208 x 64	-> 208 x 208 x 64 0.354 BF
3 route	1			-> 208 x 208 x 64
4 conv	64	1 x 1/ 1	208 x 208 x 64	-> 208 x 208 x 64 0.354 BF
5 conv	32	1 x 1/ 1	208 x 208 x 64	-> 208 x 208 x 32 0.177 BF
6 conv	64	3 x 3/ 1	208 x 208 x 32	-> 208 x 208 x 64 1.595 BF
7 Shortcut Layer: 4, wt = 0, wn = 0, outputs: 208 x 208 x 64 0.003 BF				
8 conv	64	1 x 1/ 1	208 x 208 x 64	-> 208 x 208 x 64 0.354 BF
9 route	8 2			-> 208 x 208 x 128
10 conv	64	1 x 1/ 1	208 x 208 x 128	-> 208 x 208 x 64 0.709 BF
11 conv	128	3 x 3/ 2	208 x 208 x 64	-> 104 x 104 x 128 1.595 BF
12 conv	64	1 x 1/ 1	104 x 104 x 128	-> 104 x 104 x 64 0.177 BF
13 route	11			-> 104 x 104 x 128
14 conv	64	1 x 1/ 1	104 x 104 x 128	-> 104 x 104 x 64 0.177 BF
15 conv	64	1 x 1/ 1	104 x 104 x 64	-> 104 x 104 x 64 0.089 BF
16 conv	64	3 x 3/ 1	104 x 104 x 64	-> 104 x 104 x 64 0.797 BF
17 Shortcut Layer: 14, wt = 0, wn = 0, outputs: 104 x 104 x 64 0.001 BF				
18 conv	64	1 x 1/ 1	104 x 104 x 64	-> 104 x 104 x 64 0.089 BF
19 conv	64	3 x 3/ 1	104 x 104 x 64	-> 104 x 104 x 64 0.797 BF
20 Shortcut Layer: 17, wt = 0, wn = 0, outputs: 104 x 104 x 64 0.001 BF				
21 conv	64	1 x 1/ 1	104 x 104 x 64	-> 104 x 104 x 64 0.089 BF
22 route	21 12			-> 104 x 104 x 128
23 conv	128	1 x 1/ 1	104 x 104 x 128	-> 104 x 104 x 128 0.354 BF
24 conv	256	3 x 3/ 2	104 x 104 x 128	-> 52 x 52 x 256 1.595 BF
25 conv	128	1 x 1/ 1	52 x 52 x 256	-> 52 x 52 x 128 0.177 BF
26 route	24			-> 52 x 52 x 256
27 conv	128	1 x 1/ 1	52 x 52 x 256	-> 52 x 52 x 128 0.177 BF
28 conv	128	1 x 1/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.089 BF
29 conv	128	3 x 3/ 1	52 x 52 x 128	-> 52 x 52 x 128 0.797 BF
30 Shortcut Layer: 27, wt = 0, wn = 0, outputs: 52 x 52 x 128 0.000 BF				

CUDA-version: 11010 (11020), cuDNN: 7.6.5, CUDNN_HALF=1, GPU count: 1
OpenCV version: 3.2.0
0 : compute_capability = 750, cudnn_half = 1, GPU: Tesla T4

Table 4: Results

[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
Total BFLOPS 59.628
avg_outputs = 490961
Accuracy: 98%

Reference:

[1] Wong, Yan Chiew. “DEEP LEARNING BASED RACING BIB NUMBER DETECTION AND RECOGNITION.” *Jordanian Journal of Computers and Information Technology* (2019): n. pag.

[2] Son et al, Changeui et al. “Deep Learning–based Number Detection and Recognition for Gas Meter Reading.” *IEIE Transactions on Smart Processing & Computing* (2019): n. pag.

[3] Rayson Laroca, Victor Barroso, Matheus A. Diniz, Gabriel R. Gonçalves, William R. Schwartz, David Menotti, "Convolutional neural networks for automatic meter reading," J. Electron. Imag. 28(1) 013023 (5 February 2019)

[4] R. Laroca, A. B. Araujo, L. A. Zanlorensi, E. C. De Almeida and D. Menotti, "Towards Image-Based Automatic Meter Reading in Unconstrained Scenarios: A Robust and Efficient Approach," in *IEEE Access*, vol. 9, pp. 67569-67584, 2021, doi: 10.1109/ACCESS.2021.3077415.

[5] C. Li, Y. Su, R. Yuan, D. Chu, and J. Zhu, “Light-weight spliced convolution network-based automatic water meter reading in smart city,” IEEE Access, vol. 7, pp. 174 359–174 367, 2019.

[6] C. S. Marques et al., “Image-based electric consumption recognition via multitask learning,” in Brazilian Conference on Intelligent Systems, 2019, pp. 419–424.

[7] K. Kosˇcevi ˇ c and M. Suba ´ siˇ c, “Automatic visual reading of meters using deep ´ learning,” in Croatian Computer Vision Workshop, 2018, pp. 1–6.

[8] Tsai, T. D. Shou, S. Chen, and J. Hsieh, “Use SSD to detect the digital region in electricity meter,” in International Conference on Machine Learning and Cybernetics, 2019, pp. 1–7.

```
   # layer  filters   size              input                output
   0 conv     32      3 x 3/ 1   416 x 416 x   3 ->  416 x 416 x  32 0.299 BF
   1 conv     64      3 x 3/ 2   416 x 416 x  32 ->  208 x 208 x  64 1.595 BF
   2 conv     64      1 x 1/ 1   208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   3 route  1                                     ->  208 x 208 x  64
   4 conv     64      1 x 1/ 1   208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   5 conv     32      1 x 1/ 1   208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
   6 conv     64      3 x 3/ 1   208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
   7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 208 x 208 x  64 0.003 BF
   8 conv     64      1 x 1/ 1   208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   9 route  8 2                                   ->  208 x 208 x 128
  10 conv     64      1 x 1/ 1   208 x 208 x 128 ->  208 x 208 x  64 0.709 BF
  11 conv    128      3 x 3/ 2   208 x 208 x  64 ->  104 x 104 x 128 1.595 BF
  12 conv     64      1 x 1/ 1   104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
  13 route  11                                    ->  104 x 104 x 128
  14 conv     64      1 x 1/ 1   104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
  15 conv     64      1 x 1/ 1   104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
  16 conv     64      3 x 3/ 1   104 x 104 x  64 ->  104 x 104 x  64 0.797 BF
  17 Shortcut Layer: 14,  wt = 0, wn = 0, outputs: 104 x 104 x  64 0.001 BF
  18 conv     64      1 x 1/ 1   104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
  19 conv     64      3 x 3/ 1   104 x 104 x  64 ->  104 x 104 x  64 0.797 BF
  20 Shortcut Layer: 17,  wt = 0, wn = 0, outputs: 104 x 104 x  64 0.001 BF
  21 conv     64      1 x 1/ 1   104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
  22 route  21 12                                 ->  104 x 104 x 128
  23 conv    128      1 x 1/ 1   104 x 104 x 128 ->  104 x 104 x 128 0.354 BF
  24 conv    256      3 x 3/ 2   104 x 104 x 128 ->   52 x  52 x 256 1.595 BF
  25 conv    128      1 x 1/ 1    52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  26 route  24                                    ->   52 x  52 x 256
  27 conv    128      1 x 1/ 1    52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  28 conv    128      1 x 1/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.089 BF
  29 conv    128      3 x 3/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.797 BF
  30 Shortcut Layer: 27,  wt = 0, wn = 0, outputs:  52 x  52 x 128 0.000 BF
  31 conv    128      1 x 1/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.089 BF
  32 conv    128      3 x 3/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.797 BF
  33 Shortcut Layer: 30,  wt = 0, wn = 0, outputs:  52 x  52 x 128 0.000 BF
  34 conv    128      1 x 1/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.089 BF
  35 conv    128      3 x 3/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.797 BF
  36 Shortcut Layer: 33,  wt = 0, wn = 0, outputs:  52 x  52 x 128 0.000 BF
  37 conv    128      1 x 1/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.089 BF
  38 conv    128      3 x 3/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.797 BF
  39 Shortcut Layer: 36,  wt = 0, wn = 0, outputs:  52 x  52 x 128 0.000 BF
  40 conv    128      1 x 1/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.089 BF
  41 conv    128      3 x 3/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.797 BF
  42 Shortcut Layer: 39,  wt = 0, wn = 0, outputs:  52 x  52 x 128 0.000 BF
  43 conv    128      1 x 1/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.089 BF
  44 conv    128      3 x 3/ 1    52 x  52 x 128 ->   52 x  52 x 128 0.797 BF
  45 Shortcut Layer: 42,  wt = 0, wn = 0, outputs:  52 x  52 x 128 0.000 BF
```

CUDA-version: 11010 (11020), cuDNN: 7.6.5,
CUDNN_HALF=1, GPU count: 1
OpenCV version: 3.2.0
0 : compute_capability = 370, cudnn_half = 0, GPU: Tesla K80