Assessing the YOLO Series Through Empirical Analysis on the KITTI Dataset for Autonomous Driving

Filipa Ramos[1(✉)], Alexandre Correia[1], and Rosaldo J. F. Rossetti[2]

[1] Bosch Car Multimedia S.A., 4705-820 Braga, Portugal
{Filipa.Ramos, Alexandre.Correia}@pt.bosch.com
[2] Artificial Intelligence and Computer Science Lab,
Department of Informatics Engineering, Faculty of Engineering,
University of Porto, 4200-465 Porto, Portugal
rossetti@fe.up.pt

Abstract. Computer vision and deep learning have been widely popularised on the turn of the 21^{st} century. On the centre of its applications we find autonomous driving. As this challenge becomes a racing platform for all companies, both directly and indirectly involved with transportation systems, it is only pertinent to evaluate exactly how some generic, state-of-the-art models can perform on datasets specifically built for autonomous driving research. With this purpose, this article aims at directly studying the evolution of the YOLO (You Only Look Once) model since its first implementation until the most recent version 3. Experiences carried out on the respected and acknowledged driving dataset and benchmark known as KITTI Vision Benchmark enable direct comparison between the newest updated version and its predecessor. Results show how the two versions of the model have a performance gap whilst being tested on the same dataset and using a similar configuration setup. YOLO version 3 shows its renewed boost in accuracy whilst dropping minimally on detection speed. Some conclusions on the applicability of models such as this to a real-world scenario are drawn so as to predict the direction of research in the area of autonomous driving.

Keywords: YOLO · Deep learning · Autonomous driving · KITTI Vision Benchmark

1 Introduction

Autonomous driving has established itself as the topic of the future. Most international news outlets are filled with information on research and tests being carried out on such vehicles operating in innumerable urban environments. These scenarios have already stirred much confusion and created barriers for further research. Indeed, failures of such systems are identified in tragic ways that are

A. L. Martins et al. (Eds.): INTSYS 2019, LNICST 310, pp. 203–218, 2020.
https://doi.org/10.1007/978-3-030-38822-5_14

counterproductive towards the full implementation of autonomous vehicles on real-life roads, while ensuring safety and public acceptance.

The idea of autonomous vehicles entails many complex requirements that need to be fulfilled. It certainly starts with the vehicle's perception of the surrounding environment. If the transportation system does not have a good grasp of its outside system, there is no possibility of ensuring safety. On this topic, many non-trivial tasks unfold such as object detection and tracking, traffic signs interpretation, lane detection and many more. All of these are tasks that are not fully computationally solved as of today. As such, autonomous driving imposes invaluable research opportunities and challenges for the technological world.

Specifically on RGB camera detection, there is also the impingement of hardware limitations. As a camera captures almost the same information a human eye would, it also suffers in worse scenarios such as darkness, storms, rain showers and many others in which even the human eye has difficulties detecting objects. These are frontiers hard to overcome and with still much exploration to be made as reporeted in previous studies [14,15].

Furthermore, if computational models are to be applied to these vehicles in real time, real-world ensembles, the speed of inference and reaction of the independent intelligence centre of the system is paramount. Accidents on the roads happen in a fraction of second and are many times triggered by the delay between the tasks of object detection and classification, risk assessment and corresponding body movement performed by humans when they drive any vehicle.

Broadly on the topic of computer vision and two dimensional object detection, one model that has been keeping up with the state of the art over its improvements and evolution is the YOLO model. Standing for You Only Look Once, this neural network is able to detect and classify objects on camera images. When it launched, the model devised by Redmon et al. [17] showed that real-time, unified object detection and classification was possible. YOLO was one of the first detectors that only ran through the image once, detecting and classifying all objects within this single step. This enabled the model to outperform every other alternatives in the indispensable metric of detection speed. Supporting this, its accuracy was approximately parallel to that of much slower running models.

The research for object detection has gone through an exponential growth in which many models surfaced namely the Faster R-CNN [21] and others in the search for a valid solution. In order to keep up with the evolution in the field, the YOLO version 2 [18] was born, presenting many elements of significant improvement. With these upgrades, YOLO once more proved to be an already mature, strong detector and classifier even if it still had more room for improvement. However, as it becomes more and more of a huge public topic of research, new models keep rising almost every day and most recently, Facebook AI Research (FAIR) launched its Retinanet [11], based on the use of focal loss in order to diminish more effectively the effects of class imbalance. In the current research scenario, this model is considered one of the leading references in both accuracy and speed. Even more fresh, the newest update for YOLO has launched, namely

version 3 [20]. The paper describing YOLO version 3 reports an accuracy parallel to that of the commonly found best detectors on the .50 IoU mAP metric (even Retinanet). Unfortunately and opposed to this improvement, the model has become more heavy and slower at performing detection.

On an ensemble system that draws information over a panoply of sensors, this model can certainly be a good candidate for real-world autonomous driving applications. In order to dwell in more depth how this new model can really achieve such impressive results, this article reflects on the distance spanning YOLO version 2 and the newest updated version 3, all the same applying it to a prestigious urban driving dataset in several configurations in order to ascertain how the improvements have reflected on its application.

On the topic of the dataset, the KITTI Vision Benchmark Suite [5] aims at connecting research with the real-world application which makes it ideal in order to predict more informed data on how researched models will perform outside of the laboratory. Sporting scenarios from highways to urban and rural areas of Karlsruhe, frames can contain up to 15 cars and 30 pedestrians either truncated, occluded or fully visible which makes it a dataset with abundance of easy, medium and hard cases. Many respected models with good results on other benchmarks have been found to have a poor performance on this benchmark comparatively. This paper also intends to demonstrate how YOLO – which has been evaluated on general object detection datasets like PASCAL VOC [2] and COCO [12] – can cope with the complex and challenging autonomous driving scenario.

2 State of the Art

Ever since deep learning established itself as the path to follow for object detection, the evolution on the field of generally referred to as two dimensional object detection has been characterised by two main different approaches. These methodologies seem to divide themselves in two stage detectors, which seem to prime in accuracy and the one stage detectors which are usually the fastest.

The concept of two stage detectors really kicked off with the surface of the R-CNN [7] (Regions with CNN features) model. What this model had in accuracy, it did not have in detection speed which invalidated it for such scenarios as autonomous transportation systems. In fact, the original system took over 40 seconds to perform detection on a single image. Over the years, many updates were proposed namely the R-CNN minus r [9], Fast R-CNN [6], the Faster R-CNN [21] and most currently the Mask R-CNN [8] specialised in object instance segmentation. Lenc et al. even proposed R-CNN minus r [9] which used static bounding box proposals instead of the selective search [24]. Furthermore, through further improvements made on R-CNN, specifically from Faster R-CNN [21], the selective search [24] ended up being substituted for a fully qualified region proposal neural network. Selective search proposals were estimated to take around 2 s per image which invalidates its application on a real-time scenario. Incrementally, the updates that the model suffered unified the network, reducing the strenuous

process of having to train or tune independently the feature extraction convolutional neural network, the box scoring SVM, the bounding box adjustment linear model and the non-max suppression step. Unifying this pipeline meant the network became much faster, however, its speed is still incomparable to that of the current YOLO model.

On the topic of one stage detection, the field was dominated by such models as SSD [13] and YOLO [20] for quite some time. The SSD, Single Shot MultiBox Detector, relied on loosing the feature re-sampling step and the evaluation of default boxes for feature maps of different sizes. This model surpassed, at the time, its competitor YOLO version 1. Following SSD, Fu et al proposed DSSD [4] which introduced feed forward modules with deconvolutional layers which improved accuracy over the original model. However, the contribution did not seem to be significant in terms of detection speed, having approximately equal and at some resolutions even worse detection times than its predecessor.

Recently as well, Facebook's FAIR proposed a novel network, the Retinanet which has been performing outstandingly well in both accuracy and speed. This network is inserted in that of 1 stage detectors, however, it uses a new balanced focal loss which reduces the impact of imbalance between foreground and background classes on the data. The Retinanet detector's backbone is based on the feature pyramid network [10] which has established itself as a strong architecture choice on the field of deep learning for object detection and convolutional related tasks. This network uses an hierarchy of pyramidal features each merged and combined following a top-down order. The existence of both lateral and top down connections in between feature maps enables the network to be more accurate and fast at the same time.

On the other hand, YOLO has been one of the longest standing one stage detectors. The fact that YOLO's premise focus so specifically in detection speed makes it ideal to use in autonomous driving systems. Moreover, with the evolution of the network, the trade off between its performance in terms of results and detection time is even smaller. Hence the importance of studying its behaviour in urban scenarios specially to ascertain if and how the newest version outperforms its predecessor as advertised.

3 KITTI Dataset for Object Detection

The KITTI dataset used for training and testing the models scoped by this paper is described in a work by Geiger et al. [5] in more detail. This section sums up on some general information that may be relevant for the experiences presented below.

The KITTI dataset was collected using a test vehicle equipped with a panoply of cameras (both RGB and grayscale), a laserscanner, one inertial navigation system and varifocal lenses. The raw dataset contains all the information mustered through the movement inside urban areas. 3D tracklets are provided for all labelled classes. The labelled classes that can be found in this dataset are explained below.

Car	Any standard vehicle.
Van	All kinds of vehicles that present an intermediate size and shape in between that of a Car and a Truck.
Truck	The largest kind of moving vehicles.
Pedestrian	People in movement or in position of initiating movement.
Person (sitting)	People that do not seem to be in movement on the scenario. For example, people on a park bench.
Cyclist	Any moving person on a bicycle.
Tram	A standard, urban tram.
Misc	Other kinds of objects attached to vehicles such as trailers or segway.

The raw dataset provided by KITTI explores a variety of urban scenarios. Some statistics on the tracklets can be found in Fig. 1. As we can denote on the left graph, the most frequent label is Car, followed by Van. Classes such as Truck, Pedestrian, Cyclist and Misc have approximated frequency of labels. The rare classes are Tram and Person (sitting) with special emphasis on the last as it is particular hard to find examples of this behaviour on the dataset. The graph on the right explores the frequency of tracklets per frame. From what can be seen, the largest percentage of frames seem to have around 2 and 6 tracklets.

In order to evaluate identification on the KITTI Vision Benchmark Suite, the objects are separated according to their difficulty of identification. Levels of difficulty are extracted according to the occlusion and truncation of the object to be found on the tracklet. The three levels span from **Easy** and **Moderate** to **Hard**.

The 2D labelled dataset available in KITTI's 2D object detection benchmark page contains labels for all these classes. Benchmark evaluation is done over three levels of difficulty for classes *Car* which requires an overlap of 70% with the ground truth, *Cyclist* and *Pedestrian* which require an overlap of 50%.

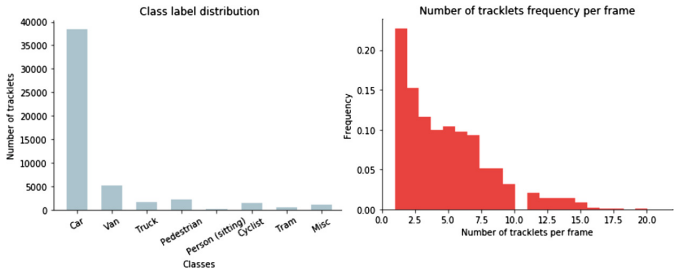

Fig. 1. Statistics on the KITTI raw data.

4 "You Only Look Once"

The You Only Look Once model looks at the full image once. This was the original premise and with this, YOLO was able to efficiently extract context from

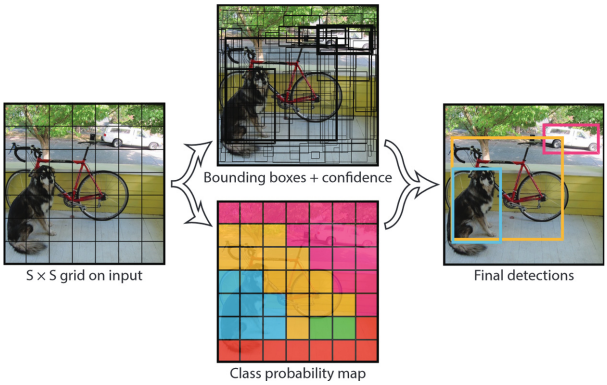

Fig. 2. From the You Only Look Once: Unified, Real-Time Object Detection [17] paper, YOLO detection pipeline.

the whole consumed image, joining both the global features and each object's specific characteristics. Taken from its original paper [17], image 2 shows the pipeline proposed initially for object detection.

As one can denote in Fig. 2, the pipeline starts from the division of the image into an $S \times S$ grid. Each cell will then predict B bounding boxes along with a confidence score for each of them. A box confidence is evaluated over the multiple classes conditional probabilities and the overall intersection over union of the predicted box with the ground truth one. All classes are predicted at the same time for the whole image and that is the premise that makes the model as fast as it has always been. The resulting tensor encodes both the number of grids, the number of classes to classify and the number of bounding boxes that each cell predicts.

A cell on the grid is responsible for a detection and classification if the object's centre is found within it. A bounding box is then represented by 5 different variables: (x, y) is the tuple that represents the centre coordinates of the box, the (w, h) contain the width and height of the box according to the image proportion and a confidence value that the box contains the object. As such, the loss function is modelled over these 5 parameters, being a combination of the sum of squared errors with some characteristics added in order to balance errors in large or small boxes and distinguish between classification and localisation mistakes.

The whole YOLO model is implemented using the Darknet framework [19] and its classifier was previously trained on Imagenet [23]. The architecture is quite simple, being a standard convolutional network with 24 layers followed by 2 fully connected layers. Non maximum suppression is also used in order to eliminate duplicated boxes.

Just like any other model, YOLO has its strengths and weaknesses. The incremental work done with the updates was in the sense of reducing the handicaps and augment even more the existent strengths. The most flagrant obstacle to tackle is the reduction of the sources of error from wrong localisation. In a

comparison in between Fast R-CNN and YOLO, Redmon et al. reported that whilst their network makes much less false positive classifications coming from background image areas, it looses most of its mAP on localisation errors. Moreover, detection of small objects specially grouped together has been a great challenge to overcome.

4.1 Version 2

The 2016 paper [18] introduced the second incremental version of the model alongside a joint training algorithm that enables training on both detection and classification data namely YOLO9000.

On the newest architecture, some changes were made which focused directly on the region proposal setup, diminishing the number of location errors and the low recall achieved by its predecessor. All the while, the objective was to always maintain the classifier's accuracy. One direct point of approach was the difficulty that the original YOLO model had with small objects.

The incremental approaches taken are described below.

Batch Normalisation. Its application on all convolutional layers improved mAP and stabilised the model. It also eliminated the need for dropout.

Higher Resolution. The classifier is tuned for 448×448 input resolution on ImageNet which helps adjust the filters for higher resolutions on the detection step. Opposed to the initial model which trained the classifier at a 224×224 resolution and used 448×448 for detection.

Anchors. Inspired by Faster R-CNN's offset prediction, anchor boxes are used. Network resolution is reduced to 416×416 in order to have only one cell at the centre of the image. Higher resolution output after the removal of one pooling layer.

Clustering. Anchor priors are chosen through $k-$means clustering. With a $\mathrm{k} = 5$, anchors are defined using a distance metric that involves the intersection of union between the box and its centroid.

Location Predictions. Bounding boxes are predicted according to the responsible cell. Sigmoid activation is used in order to constraint the ground truth to a range between 0 and 1. Parameters t_x, t_y, t_w, t_h, t_o are used to calculate the boxes centre/dimensions alongside with the top left corner offset coordinates (c_x, c_y) and the prior width and height p_w, p_h.

Fine-Grained Features. Additional feature map of larger resolution (26×26) stacked with the 13×13 feature map through a pass-through layer.

Multi-Scale Training. Training done with random net resolutions switched at each 10 batches.

Darknet-19. Classifier network changes to Darknet-19, a more mature and evolved model with 19 convolutional layers and 5 max pooling layers. Data augmentation, hue, saturation, crops, rotations and exposure shifts used in order to introduce variety in the data.

4.2 Version 3

Most recently, the YOLO project web page[1] has launched a newer update – version 3. On a preprint arXiv archive the changes are described and the newer results are advertised as still being much faster than any model and on par with accuracy on the AP .50 metric. However, on the newer .95 metric, the model can not keep up with Retinanet in terms of accuracy, having around 7% less mAP. This fact immediately leads us to the conclusion that this model seems to be great at localising and classifying objects but not great at giving an exact, almost perfect bounding box that contains the object.

Objectness. Objectness score was introduced in version 2. However, on the newest update, this score is calculated using logistic regression which represents the ratio of overlapping in between the ground truth object and the bounding boxes prior.

Classification. Softmax is removed and independent logistic classifiers are added instead. Predictions are then obtained using binary cross-entropy.

Pyramid Extraction. Much like feature pyramid networks [11], YOLOv3 extracts features at several scales and concatenates the upsampled feature maps with ones obtained on earlier stages.

Anchors. Box priors are still extracted through $k-$means clustering, only now with 9 clusters and 3 arbitrary scales.

Darknet-53. The classifying network evolves towards 53 layers and adds shortcut connections in order to process the connected feature maps with more fine grained information. This network is also trained on ImageNet [23] and achieves top classification on par with ResNet-152 but much faster.

With these newer updates, the model moves towards more accuracy. Even whilst loosing some speed, it is still a very powerful and fast network. It is reported that small objects are not such a challenge for the model anymore however, it seems that more difficulties arise on the larger and medium objects.

5 Experiments

In order to ascertain how the model would perform on the KITTI dataset, several experiments were conducted. The bulk of experiences were carried out using the raw dataset available at KITTI's website[2] since the objective of this work was not to benchmark the YOLO series but rather to observe its performance under different conditions and versions. The dataset used contains around 12932 different images. Since this collection encompasses all frames taken during the trips made with the test vehicle, several consecutive images are very similar with small changes. A validation dataset was drawn from 355 frames. The validation

[1] Found at https://pjreddie.com/darknet/yolo/.
[2] Download at http://www.cvlibs.net/datasets/kitti/raw_data.php.

set always contains frames from different drives since inside the same drive the variation of data is very small. Even though the validation set is small compared to the number of existent frames, the data extracted is very significant since it eliminates any kind of repetition and similarity in between most of the frames from training.

All experiments described below were run on a single NVIDIA Geforce GTX 1080 Ti with 11GB of dedicated memory. Libraries CUDA and CudNN are used on version 9.0 and 7.0 respectively. Also, all models were trained until they stabilised and reached a convergence stage. The weights chosen for each result comparison are those at the best early stopping point in order to avoid overfitting. Loss values displayed are taken from an average around the early stopping point. Val time corresponds to the time that takes to validate performance on the whole testing dataset (which means detection time for 355 images). The threshold used for mAP calculations is 0.25. Training thresh equals 0.6.

Experiments follow an incremental evolution pattern. In fact, if a parameter is found to be better than the others in a given experiment, in following experiences these best performing configurations are kept, varying only the tweaks in study.

5.1 RGB

All training was carried out using the colour left images of the dataset from all available drives and frames with tracklet information.

Network Resolution. A study that was found to be interesting was the variation of the network resolution. Since the input images have a rectangular resolution (width larger than height by approximately 3.6 times), when training was deployed on squared resolutions, the end results seemed to be much lower. For a squared resolution test, the 416×416 size was chosen. Since resolutions over 832 are not advisable, the testing rectangular resolution 256×832 was used. This is approximately a factor of 1.5 (closer resolution multiple of 32 as expected by the model) times smaller than the original dataset image size.

Anchors. YOLO comes with default anchors. However, for each dataset, using the specified $k-$means approach for each version, it is possible to generate new anchors that are supposedly more appropriate for the training in question. Training was carried out for both versions with the default priors and with calculated ones. The results can be observed in Table 2.

Random. One of the new introductions made by the YOLOv2 paper [18] was the enabling of the model to switch its own resolution at random in order to enable the model to generalise better for other input dimensions. Tests with this configuration were also carried out. The comparison is made between random training runs, starting either with a squared or a rectangular resolution for both models.

Table 1. Results obtained for the variation of the network resolution. The evaluation metrics are mAP, which stands for mean average precision, and IOU, which represents the intersection of union.

Network	Resolution	Loss	mAP	IOU	Val time
YOLOv2	416 × 416	0.30	56.06	55.05%	4 s
YOLOv3	416 × 416	0.9	72.58	65.72%	8 s
YOLOv2	256 × 832	0.28	69.91	59.27%	4 s
YOLOv3	256 × 832	0.43	**75.51**	**73.12%**	9 s

Table 2. Results obtained for the variation of the anchors.

Network	Anchors	Loss	mAP	IOU	Val time
YOLOv2	standard	0.28	69.91	59.27%	4 s
YOLOv3	standard	0.43	**75.51**	**73.12%**	9 s
YOLOv2	generated	0.3	68.59	59.71%	4 s
YOLOv3	generated	0.56	73.48	69.56%	9 s

5.2 Grayscale

Having trained the model on RGB images and using the exact same validation set, however this time with grayscale left pictures, the model was validated. YOLO uses saturation, hue changes and other techniques referred on previous sections for data augmentation. This experiment shows how this technique can help the model generalise better to other environments. The results were quite positive and are described in Table 4. Figure 3 shows a detection with the model on a painting, as an example for other kinds of generalisation.

5.3 Version 2 vs Version 3

Wrapping up all the experiences previously exposed, the best configurations from each version are chosen in order to directly compare how both models wage on the dataset. On this dataset, the best configuration found for both models does not use random training but a rectangular resolution and the standard anchors. Table 5 shows a strong improvement on mean average precision of around 5 points and an increase for intersection of union of over 13% which are incredibly substantial results for YOLO version 3 over its predecessor.

From the bundle of experiments carried out, it is possible to conclude that, overall, the YOLO model has improved considerably. Even so, there is still a visible trade-off in the newest update such as the loss of detection speed in detriment of accuracy, which is obvious in all previous experiments. In most cases, it has approximately doubled the detection time for 355 images, properly observable in Table 5 through the column *Val time*. There certainly is a trade-off between accuracy and speed which may influence the applications of this model

Table 3. Results obtained for the variation of the random resolution switch.

Network	Random	Resolution	mAP	IOU	Val time
YOLOv2	0	256 × 832	69.91	59.27%	4 s
YOLOv2	1	416 × 416	61.05	55.59%	4 s
YOLOv2	1	256 × 832	22.21	54.27%	4 s
YOLOv3	0	256 × 832	**75.51**	**73.12%**	9 s
YOLOv3	1	416 × 416	72.58	65.72%	8 s
YOLOv3	1	256 × 832	11.42	50.13%	9 s

Table 4. Results obtained for the tests on grayscale data.

Network	mAP	IOU	Val time
YOLOv2	63.79	57.59%	**5 s**
YOLOv3	**71.45**	**70.27%**	8 s

in certain real-world situations. However, it is still the fastest model known publicly, producing detection in around 25 ms per image on the newest update at a large resolution (of 832 in one parameter). This behaviour was more than expected as the number of layers was more than doubled from Darknet-19 to Darknet-53.

As for resolution, the YOLO model shows different behaviours in version 2 and in the newest update. It seems reasonable to conclude that keeping the aspect ratio of the original training images is the best approach regardless of the model being used. However, it is also obvious to conclude that YOLO version 2 was much more sensible to the object's original ratio. This is proved by the astounding improvement in mAP of 13.85 points gained by using a rectangular

Fig. 3. Output of YOLO version 3 on a painting that did not have a coincident resolution with the network's training resolution. Painting by Tom Brown [1], 2015.

Table 5. Compilation of the best results obtained for each version of the YOLO system.

Network	Loss	mAP	IOU	Val time
YOLOv2	0.28	69.91	59.27%	4 s
YOLOv3	0.43	**75.51**	**73.12%**	9 s

resolution as seen in Table 1. Meanwhile, the improvement in version 3 is only of around 3 points. This may be due to the use of connection of the different scales and feature maps that make it more invariable to natural scales on the newest update. The lack of sensibility in between resolutions is certainly a good aspect as it may increase the capacity of the model to generalise better in new conditions.

When it comes to the prior generation, the results seem a bit puzzling. Even if the standard anchors have been properly tuned on larger datasets such as COCO [12] and PASCAL VOC [2] with a large abundance of classes and cases, it is still surprising to find that the standard anchors produce slightly better end results for both models. Nevertheless, this is good news as it means that YOLO can be adapted to new datasets with minimal pre-processing overhead. Table 2 shows small differences between using standard and generated anchors, having only an increase around 1–2 points on mean average precision. Even though slightly, the use of generated anchors over the standard ones for version 3 seems to produce worse results compared to the behaviour of version 2.

The possibility to variate resolution during training was also tested and yielding interesting results. The use of random for training with a starting squared resolution gives good results even though the testing images are rectangular in both version 2 and 3. Even so, training for a rectangular resolution specifically proves to the best approach, assumption proved by the results showcased in Table 3.

Moreover, the use of random training from a starting rectangular resolution produces rather poor results on both metrics even if more heavily on average precision. This seems to be a limitation in the classifier. Both versions of the model have the same kind of behaviour towards this parameter. From Table 3 one can denote that having a direct rectangular resolution produces superior outcomes compared to having a randomly sized training process which consumes more gpu resources and takes around twice the time to train. When the ratio of the images is known beforehand, it is preferable to keep the network's resolution parallel to that of the original aspect whilst still keeping it under 832 for width or height since larger resolutions have a directly proportional decrease in detection speed that would nullify the objective of trading off speed over accuracy. This was also tested in extra experiments.

YOLO is able to generalise to different resolutions, environments, lightning conditions and even paintings. On the grayscale experience, with results showcased in Table 4, it is only noticeable a little drop in mean average precision compared to the colour validation set. This is an invaluable quality for an

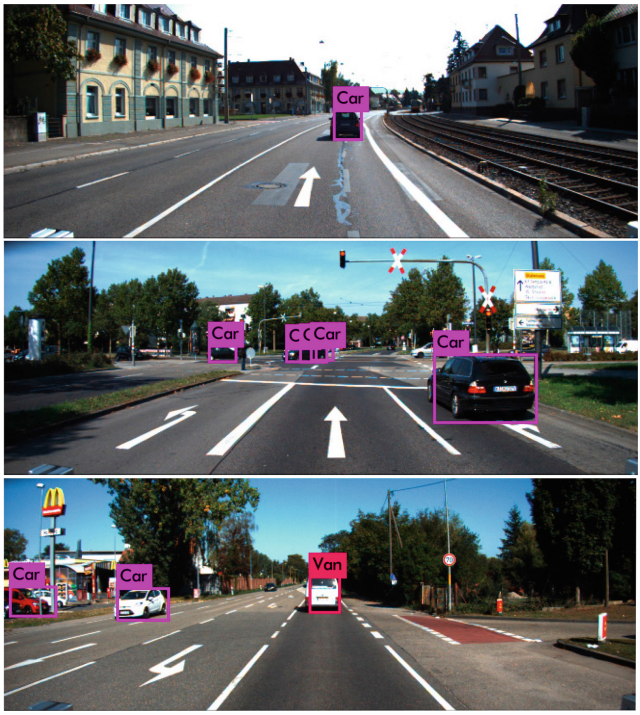

Fig. 4. Output of YOLO version 3 on the KITTI dataset for lane detection [3].

autonomous driving situation. On the road, many unplanned, new scenarios arise and the model needs to be elastic enough to respond to all of them with the degree of safety and confidence as the cases it was trained for. On this aspect, YOLO maintains and in some aspects even increments on its generalisation value. Table 4 depicts less loss in average precision in the newest version than in its predecessor whilst comparing these with the best obtained on the RGB set.

6 Conclusions

It was intuitively expected that the additional layers would decrease the speed of the network whilst boosting its performance. However, there are more tweaks on the newest version that prove that the YOLO series has even more room to grow and expand itself into a possible alternative to meet the demanding requirements of the autonomous driving scenario. However, its behaviour with the prior generation must be more deepened in order to ascertain its true advantages to the model. Using random resolution training is an expensive alternative that must only be applied to certain use cases.

Overall, YOLO version 3 is a much superior object detection model over version 2. The model has reached new heights in accuracy in all configurations,

keeping a standard mean average precision around 70% which was rare and quite hard to obtain by using version 2 on this dataset. The improvements are expressed as well on the placement of more correct bounding boxes, supported by the increasing intersection of union.

Moreover, the fact that the YOLO series is able to generalise so well to grayscale images is a step forward towards solving the problem of object identification in such complex environments as night time, rain conditions, storms and such. If the model were to show such results on applications of thermal images, it could be a reliable solution for the more demanding conditions to be faced on the road. Furthermore, if YOLO could perform detections on these types of cameras whilst only having been trained on RGB images, the gathering of such powerful, mixed information would be a gigantic step for autonomous driving as it would allow having less sensors on the vehicles and less training processes in order to make them operational. Very much similarly to YOLO9000 [18] that generalised to 9000 classes, future work on the direction of enabling YOLO to perform just as accurately on several types of cameras could be an interesting step moving forward. Another study of great interest would be the thorough analysis of the relationship between the number of layers in the classifier network and its impact on speed degradation, so as to identify the optimal point for the reasonable trade-off between accuracy and speed.

Finally, this study contributes with a thorough analysis of YOLO version 2 and version 3, emphasising on the improvements of the latter version towards supporting the autonomous driving scenarios. Currently YOLO is underlying the development of an Artificial Transportation Systems platform, coined MAS-Ter Lab [22], endowing such an environment with computer-vision-as-a-service functionality. Both traffic and transport control and management, as well as the simulation of autonomous vehicles in urban settings [16] are under development within the MAS-Ter Lab platform using the YOLO model.

Aknowledgements. This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project n° 037902; Funding Reference: POCI-01-0247-FEDER-037902].

References

1. Brown, T.: Plein Air Oil Painting (2015). http://tombrownfineart.blogspot.com/2015/06/25-cars-8x10-plein-air-oil-painting-by.html
2. Everingham, M., Eslami, S.M.A., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL visual object classes challenge: a retrospective. Int. J. Comput. Vis. (2014). https://doi.org/10.1007/s11263-014-0733-5
3. Fritsch, J., Kuehnl, T., Geiger, A.: A new performance measure and evaluation benchmark for road detection algorithms. In: International Conference on Intelligent Transportation Systems (ITSC) (2013)
4. Fu, C.Y., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: DSSD: deconvolutional single shot detector. CoRR abs/1701.06659 (2017)

5. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the KITTI dataset. Int. J. Robot. Res. **32**(11), 1231–1237 (2013). https://doi.org/10.1177/0278364913491297
6. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision 2015 Inter, pp. 1440–1448 (2015). https://doi.org/10.1109/ICCV.2015.169
7. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014). https://doi.org/10.1109/CVPR.2014.81
8. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision 2017, pp. 2980–2988, October 2017. https://doi.org/10.1109/ICCV.2017.322
9. Lenc, K., Vedaldi, A.: R-CNN minus R. In: British Machine Vision Conference (2015)
10. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 936–944, July 2017. https://doi.org/10.1109/CVPR.2017.106
11. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision 2017, pp. 2999–3007, October 2017. https://doi.org/10.1109/ICCV.2017.324
12. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
13. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
14. Loureiro, P.F.Q., Rossetti, R.J.F., Braga, R.A.M.: Video processing techniques for traffic information acquisition using uncontrolled video streams. In: 2009 12th International IEEE Conference on Intelligent Transportation Systems, pp. 1–7, October 2009
15. Neto, J., Santos, D., Rossetti, R.J.F.: Computer-vision-based surveillance of intelligent transportation systems. In: 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–5, June 2018. https://doi.org/10.23919/CISTI.2018.8399240
16. Pereira, J.L.F., Rossetti, R.J.F.: An integrated architecture for autonomous vehicles simulation. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC 2012, pp. 286–292. ACM, New York (2012)
17. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788, June 2016. https://doi.org/10.1109/CVPR.2016.91
18. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6517–6525, July 2017. https://doi.org/10.1109/CVPR.2017.690
19. Redmon, J.: Darknet: Open source neural networks in C (2013–2016). https://pjreddie.com/darknet/
20. Redmon, J., Farhadi, A., Ap, C.: YOLOv3 : an incremental improvement. Technical report (2018). https://doi.org/10.1109/CVPR.2017.690

21. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031
22. Rossetti, R.J.F., Oliveira, E.C., Bazzan, A.L.C.: Towards a specification of a framework for sustainable transportation analysis. In: 13th Portuguese Conference on Artificial Intelligence, EPIA, Guimarães, Portugal, pp. 179–190. APPIA (2007)
23. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
24. Uijlings, J.R., Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. Int. J. Comput. Vis. **104**(2), 154–171 (2013). https://doi.org/10.1007/s11263-013-0620-5