A DEEP LEARNING APPROACH TO AUTONOMOUS DRIVING IN URBAN ENVIRONMENT

Paul DIACONESCU[1], Victor-Emil NEAGOE[2]

This paper is dedicated to road object detection in urban environment using Deep Learning Neural Networks (DLNN). We have chosen the advanced architecture YOLOv5 trained on COCO 2017 dataset (containing 80 classes of objects in 1.5 million images) and we have specialized YOLOv5 to recognize objects from BDD100K dataset (Berkeley Diverse Driving Video Database, including 100,000 images). We have kept eight classes from BDD100K: car, truck, person, traffic sign, traffic light, bus, rider and bike.

The object detection performance (average precision) obtained in this paper for BDD100K dataset is the highest, compared with previous published approaches for the same dataset, and the highest for the mapping of YOLOv5 and BDD100K dataset.

The research results prove that one can apply state of the art cutting edge deep learning technologies to open new perspectives for automotive industry.

Keywords: pattern recognition, autonomous navigation, deep learning neural networks (DLNN), transfer learning, YOLOv5, COCO 2017, BDD100K

1. Introduction

The automotive industry has a high bar in terms of quality and reliability for the components it uses. While reducing the huge associated costs and time to market, the automakers have to adopt innovative techniques that allow them to differentiate. Published research [1][2] tracked the automated car concept back in 1920, and significantly evolving after 1980 with Carnegie Mellon University (CMU) Navlab and ALV projects. CMU had produced the first automated car in 1989, called ALVINN, a car that used a neural network to drive. These days, advanced systems are able to perform automated driving in diverse and complex conditions [3][4].

The automotive industry has defined six levels of autonomy for cars, starting from Level 0 (no autonomy) to Level 5 (full autonomy). At this moment, the most advanced systems released in production can be matched to an area around Level 3 (Conditional Automation) where we can talk about autonomous

[1] PhD Student, Dept. of Applied Electronics and Information Engineering, University POLITEHNICA of Bucharest, Romania, e-mail: paul.diaconescu@gmail.com
[2] Prof. PhD, Dept. of Applied Electronics and Information Engineering, University POLITEHNICA of Bucharest, Romania, e-mail: victoremil@gmail.com

vehicles that drive themselves while still requiring a human driver behind the wheel for the most important decisions. At this level and above, the vehicles need to be able to "read" and adapt to the environment. Part of the understanding of the environment is the detection of relevant traffic objects: cars, people, traffic lights, motorcycles, traffic signs.

One set of machine learning techniques with excellent results in all computer vision domains is Deep Learning. Diverse efforts [5][6][7] had been already made to use its power in automotive field.

As previously presented in [8], Deep Learning approaches for autonomous driving has been split into: mediated perception (uses the driving markers in the image to create a world representation surrounding the car, for example items such as traffic signs and obstacles in the road are classified to determine a driving action) and behavior reflex (directly maps the input data to a driving action).

Building on previous research directed to machine learning for pedestrian detection [9][10] and neural networks for autonomous navigation [11][12], in this paper we are adding an additional contribution to the automotive field developments by introducing a novel technique of mapping object detection technologies to an automotive environment.

2. Transfer learning

Humans can use learnings from past experiences in new ways. They don't need to restart learning from scratch. Depending on the similarity of the experiences, they can reuse more or less from their past learnings. For example, for a person acknowledged with driving a car, driving a truck is relatively easy, while for driving a plane, the initial car driving knowledge has a very low relevance.

Sometimes, previous learnings can help in new endeavors as for example learning a programming language can benefit from knowing more than one programming language previously.

Machine learning methods have been created for very specialized domains, for example hyperspectral pixels classification or financial credit score estimation. Even if these specializations are very different, the machine learning methods used for different tasks in these domains have similar components that can be transferred [13] or modified to accommodate reusability in the other domain.

Specific to Deep Learning, Transfer Learning can take into consideration reusing a state-of-the-art neural network architecture, adopting components as the same method of using dynamic learning rate or even reusing entire trained network layers in a new network.

The components that are transferred can be used as they are, "frozen", or they can be modified or allowed to be modified during a specialized learning.

For learning transfer of a Deep Learning Neural Network, the reuse of layers should be first considered. Typically, initial layers are capturing simpler and more generic features while the last layers are capturing more structural or specific features. As an example, for using a DLNN trained for detecting dogs in a task of detecting cats, the entire architecture must be kept, while first two layers could be frozen and during a specialized training with cats images, the last four layers can be left to be dynamically changed by the DLNN training mechanism as presented in Fig. 1. In a special case that time is not a hard constraint and cat images are widely available, all layers could be retrained for a higher DLNN performance to be achieved.

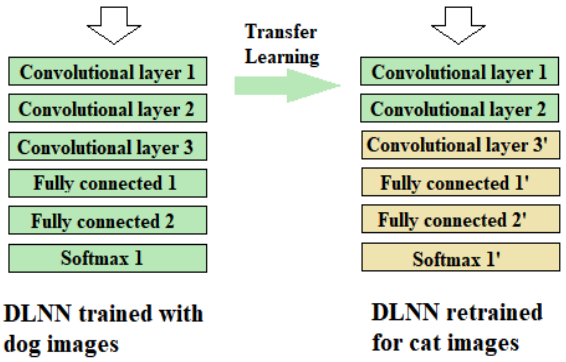

Fig. 1. Learning transfer example

This paper addresses the learning transfer from a state-of-the-art DLNN for object detection trained to detect 80 objects from COCO dataset to a DLNN specialized to detect 8 objects from an automotive related dataset.

3. YOLOv5 Deep Learning Neural Network

The most known DLNNs used for transfer learning in computer vision are Alexnet, GoogleNet, Inception [14], ResNet and VGG.

However, one of the most advanced DLNN architectures in terms of object detection accuracy, fast response and reduced network size is YOLOv5 [15]. These characteristics make the network YOLOv5 great for automotive applications and in general applications where low latency of response is a must.

Having a solid method to transfer learning from world top performance DLNN to similar applications with different datasets would be a way to ensure high performance DLNNs are immediately used in production without the energy and time typically spent to train a new network from scratch.

In this paper, we have used the smallest YOLOv5 architectures, called YOLOv5s and YOLOv5s6 due to the reduced time for training compared with the other YOLOv5 architecture types. However, even the smallest YOLOv5 has a high complexity given by 191 layers and 7.5M gradients.

A previous research [16] used YOLOv3 DLNN for automated annotation of objects related to urban driving environments. The result has been that the detector was able to detect 50 pedestrians from 114, that were manually annotated by the user in the test set.

4. Proposed method

A. Transfer Learning Method

For an efficient use of YOLOv5 with a different dataset than default trained, we have tried the following techniques:

1. Training of a clean DLNN (YOLOv5 structure) without changes with the new dataset
2. Training of the already trained DLNN, with the training elements of the new dataset
3. Training of the already trained DLNN, with the training elements of the new dataset while keeping most of the layers frozen
4. Training of the already trained DLNN, with resized (at super/increased resolution) training elements of the new dataset
5. Training of the already trained DLNN, with resized (down sampling the images to be able to run faster and run more epochs) training elements of the new dataset
6. Training of the already trained DLNN, with the training elements of new dataset and changing the default optimization algorithm (Stochastic Gradient Descent) with Adaptive Moment Estimation (ADAM)
7. Alteration of the original DLNN fitness function to optimize the hyperparameters of the DLNN for training the DLNN with the training elements of the new dataset

B. Evaluation Criteria

The object detection mechanism is based on two components: object identification for a given image and prediction of its coordinates. They can be measured using the indices of Recall (how well you find all the positives) and Precision (how accurate are your coordinates predictions).

For the purpose of measuring the DLNN performance of object detection, we have used the mean Average Precision (mAP@0.5) as a main criterion as this includes the components of both Recall and Precision. The index mAP@0.5 is the

mean Average Precision (calculated as the area under the Precision vs Recall curve) at IoU > 0.50, where IoU is Intersection over Union (calculated as the intersection between the ground-truth bounding box of an object and the corresponding predicted bounding box). A value for IoU higher than 0.5 is agreed to correspond to a good detection of an object. However, the index mAP@0.5 is calculated for all objects and this is a measure of the DLNN capabilities, not a measure of how well each object has been detected.

C. **Evolving parameters with a Genetic Algorithm**

YOLOv5 includes a file for setting all hyperparameters as learning rate, momentum, weight_decay, warmup_epochs, warmup_momentum and others. While these values have been found as being the optimum values for the COCO Dataset, there is a small chance to be the optimum combination of hyperparameters also for other datasets.

In the search for the proper hyperparameters values for BDD100K dataset [17], we have used a default mechanism included in YOLOv5 release, called "Hyperparameter evolution" and we have done customizations over this mechanism.

Hyperparameters evolution is using Genetic Algorithms as crossover and mutation to update the hyperparameters in order to obtain better results for a defined objective. The hyperparameters crossover uses the crossover genetic operation model to combine the genetic information of two parents in order to obtain a new offspring. In this case the parents are chosen as the best two past results. The hyperparameters mutation uses the mutation genetic model to alter the genetic information of a parent in order to obtain a new offspring. In this implementation, the operation is used on the best past result.

60 simulations of 6 epochs have been done to find the hyperparameters combinations that produce the best index mAP@0.5, then the winners have been used for longer simulations (50 epochs or more). In 6 epochs the index mAP@0.5 has been obtained as belonging in the interval between 0.20 and 0.30.

YOLOv5 DLNN has been created with a good flexibility for specific network objectives and before training it can be adjusted to the researcher objective through a fitness function that initially is chosen as: [0.0, 0.0, 0.1, 0.9], the values representing the percent of which each of the [Precision, Recall, mAP@0.5, mAP@0.5:0.95] should be prioritized for optimization.

Hyperparametric evolution lets the researcher to define a number of tries and a number of epochs per try. Then, after each training iteration of the defined number of epochs, the Genetic Algorithm updates the hyperparameters and another run is started until the defined number of tries is reached.

The genetic algorithm is trying to optimize hyperparameters for the objectives indicated by the fitness function and is using the results obtained after each training iteration for the next genetic operations. There are a few subtleties related to the process, for example if the best mAP@0.5 can be obtained after a number of around 100 epochs, theoretically is not enough to run the hyperparameters optimization with iterations of less epochs because the DLNN will be optimized only for fast initial growth of mAP@0.5 instead of finding the global maximum mAP@0.5.

D. **Software architecture**

YOLOv5s and YOLOv5s6 (the smallest variants, release 5 [18]) DLNNs are written in Python and run with Pytorch framework. We have done our simulations under Ubuntu Linux 20.04, using Python 3.9 (beta) compiler and NVIDIA CUDA 11.2.

5. Experiments and Results

A. The Datasets

Common Objects in Context (COCO) 2017 is the default dataset used for the development of YOLOv5 DLNN. The dataset includes 80 types of objects in 1.5 million object instances.

We have used transfer learning to recognize objects from BDD100K dataset. This is a Large-scale Diverse Driving Video Database from Berkeley, including 100,000 images with good context diversity including multiple cities, multiple weathers, multiple times of day (including low light, low visibility) multiple scenes types (including overlapped objects).

(A) (B)

Fig. 2. Examples of pictures from BDD100K dataset

We have kept eight classes as car, truck, person, traffic sign, traffic light, bus, rider and bike having the distribution shown in Fig. 3.

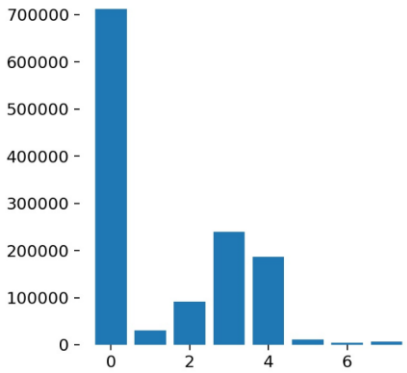

Fig. 3. The object class distribution

The dataset has an imbalanced class distribution which implies the DLNN will be better on recognizing different classes of objects that are better represented. While marginal better recognition of the most represented classes is expected, we have been aware the main risk resulted from imbalanced datasets is that DLNN will become biased toward the majority class(es) and will fail to learn what makes other classes different, predicting the majority class(es) more often, as this will not affect too much the loss function due to low representation of the other classes. The main potential issue resulting from this risk is the overfitting of the majority class, which we have verified and didn't find as significant or systemic impact after 100 epochs simulations.

We have also used a technique that has images with a high content of low-mAP@0.5 objects being selected with higher likelihood during training. Even if the technique had a (limited) positive impact on recognition of the sub-represented classes, the total mAP@0.5 result has been under our best result.

B. Transformation of annotations

When referring to an element of a dataset we refer to the image including a number of objects and to the annotation of that image, a data structure including the type of the objects and object coordinates. In special cases, additional information can be added to establish the context of each image and this information can be: date, time of day, position of the photo camera, type of the photo camera. Other information is intrinsic as the image resolution or can be further deducted from the image as in case of number of colors.

One of the issues for matching good datasets and state of the art DLNNs is the incompatibility given by the customized way the DLNNs are reading and using information from the dataset elements. Most of the times, a DLNN is developed with a single dataset and as an effect is able to read only a type of elements specific to that dataset.

To address the issue, a series of technical transformations can be made to add the annotation to the expected format by the DLNN.

The original BDD100K dataset included annotations in JSON format, which cannot be recognized by YOLOv5 DLNN (or earlier YOLO implementations).

A set of operations have been executed to transform the annotation to the required format:

- The JSON file has been read, going through all of the structured data and for each image name
- The size of all .JPG files has been determined, which is needed for YOLOv5 coordinates, because these coordinates are transformed in percent/ratio relative to total image resolution
- A new .txt file has been created for each .JPG file
- In each text file associated to an image, for each object the type has been added and the coordinates have been transformed from .JSON (in this case COCO/YOLOv3 annotation: top-left and bottom right of the box are given) to YOLOv5: Xcenter, Ycenter, weight, height of the object

The code used to automate the annotation transformation for 100,000 elements (training and validation) has been uploaded on GitHub platform and can be used as Open-Source code at: https://github.com/Delphi89/JSON2YOLOv5.

C. Experimental Results

After trying a set of techniques for the most efficient transfer learning, the proper technique for our case has been the following:

As the objective has been having a high mAP@0.5, we have tried to modify the YOLOv5 fitness function with different values. Against intuition, the best results haven't been achieved when having mAP@0.5 very high and the other objectives very low, instead the best results have been obtained when finding an equilibrium between them.

[Precision, Recall, mAP@0.5, mAP@0.5:0.95] = [0.2, 0.2, 0.4, 0.4]

A trained (trained with COCO) version of YOLOv5 DLNN has been used for a large number of training iterations (training with BDD100K) with the only scope of finding the optimum hyperparameters for training the network. This has been achieved by selecting the hyperparameters that allowed the fastest progression of mAP@0.5 after 6 epochs, then using the winner set of hyperparameters for more evolving in simulations with 50 epochs. After finding the most efficient hyperparameters in the 50 epochs contest, we have run a training iteration with enough epochs to obtain a high mAP@0.5.

Examples of images with detected objects marked, are presented in Fig. 4.

(A) (B)

Fig. 4. Object detection for the pictures from Fig. 3

The difference between an annotated image and a tested image is shown in Fig 5. The objects from the tested image have been well detected, in practice this being interpreted using a threshold dependent on the application (a higher threshold is needed in automotive field, a lower threshold for the consumer area).

(A) (B)

Fig. 5. (A) Image with original labels. (B) Image with predicted labels

The results of our most relevant simulations are presented in Table 1.

Table 1

Results

Evaluation Indicator	mAP@0.5	Training time	Average detection time per image
Reference: **Original YOLOv5s,** trained with COCO, image size 640 / **Original YOLOv5s6**, tested with COCO, image size 1280	55.4 / 61.9	N/A	0.010s / 0.024s
YOLOv5s6 trained with BDD100K for 100 epochs, image size 1280, tested with BDD100K	62.6	65.5 hours	0.024s
YOLOv5s trained with BDD100K for 100 epochs, image size 640, tested with BDD100K	59.6	18.1 hours	0.010s
YOLOv5s trained with BDD100K for 100 epochs, image size 1280, tested with BDD100K	68.7	65.2 hours	0.024s
YOLOv5s trained with BDD100K for 100 epochs with 24 layers frozen, image size 640, tested with BDD100K	19.6	17.3 hours	0.010s
YOLOv5s trained with BDD100K for 100 epochs, using ADAM instead of SGD optimization algorithm, image size 640, tested with BDD100K	30.2	17.5 hours	0.010s
Results from other papers [19] for object detection on BDD100K dataset	45.7	N/A	N/A

Results from other papers [20] for object detection with YOLO (from scratch, without transfer learning) on BDD100K dataset	18.6	N/A	N/A

We have obtained better object detection results for BDD100K dataset compared with the best results obtained by YOLOv5 with COCO dataset. The most probable explanation is that the number of object classes is lower in case of BDD100K dataset than in case of COCO. However, for a specialized application as autonomous driving in this case, the results can be considered as very good.

The latest release of YOLOv5 adds the option to use the network with live recorded videos or YouTube videos. We have used the trained network with driving videos and the real time DLNN response is excellent, one static example being presented in Fig 6.

Fig. 6. Real time prediction on driving video from YouTube

6. Conclusions

The research has successfully demonstrated that the automotive industry can reuse state of the art Deep Learning technologies and adapt them for building autonomous vehicles. Reusing cutting edge technologies ensures a lower cost and a faster time to market while opening new perspectives for automotive methods.

The presented mapping of YOLOv5 and BDD100K have obtained the best score published so far for this pair due to the learning transfer technique. This score is also the highest score obtained for object detection performance with the Berkeley Diverse Driving Video Database.

The code for automated transformation of image annotations is Open-Source and can be reused [21], reducing the cost of a similar approach.

We have shown that modifying the fitness function and using the genetic algorithms, the results can be optimized for different and specific objectives.

7. Limitations and next steps

When working with a large number of elements (images in this case) is very hard to understand all the details of the DLNN performance. We will further improve the DLNN training capabilities, adding more complex object detection performance metrics and adding automated capabilities for identifying object detection errors.

R E F E R E N C E S

[1] *A.R. Fayjie, S. Hossain, D. Oualid, and D.-J. Lee*, Driverless Car: Autonomous Driving Using Deep Reinforcement Learning in Urban Environment", in Proc. of 15th International Conference on Ubiquitous Robots (UR), Hawaii Convention Center, Hawaii, USA, June 27-30, 2018, pp. 896-901.

[2] *D. A. Pomerleau*, "Alvinn: An autonomous land vehicle in a neural network," in Advances in neural information processing systems, 1989, pp. 305–313.

[3] *Rauskolb, Fred & Berger, Kai & Lipski, Christian & Magnor, Marcus & Cornelsen, Karsten & Effertz, Jan & Form, T. & Graefe, Fabian & Ohl, Sebastian & Schumacher, Walter & Wille, Jörn-Marten & Hecker, Peter & Nothdurft, Tobias & Doering, Michael & Homeier, Kai & Morgenroth, Johannes & Wolf, Lars & Basarke, Christian & Berger, Christian & Rumpe, Bernhard.* (2009). Caroline: An Autonomously Driving Vehicle for Urban Environments. 56. 441-508. 10.1007/978-3-642-03991-1_11.

[4] *J. Guivant, E. Nebot, and S. Baiker,* "Autonomous navigation and map building using laser range sensors in outdoor applications," Journal of robotic systems, **vol. 17**, no. 10, 2000, pp. 565–583

[5] *Luckow, Andre & Cook, Matthew & Ashcraft, Nathan & Weill, Edwin & Djerekarov, Emil & Vorster, Bennie.* (2016). Deep Learning in the Automotive Industry: Applications and Tools. 3759-3768. 10.1109/BigData.2016.7841045.

[6] *Dheekonda, Raja & Panda, Sampad & Khan, Md & Hasan, Mohammad & Anwar, Sohel*. (2017). Object Detection from a Vehicle Using Deep Learning Network and Future Integration with Multi-Sensor Fusion Algorithm. 10.4271/2017-01-0117.

[7] *Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X.* Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems, 2009, 30(11), 3212-3232

[8] *N. Gallardo, N.Gamez, P. Rad and M. Jamshidi,"*Autonomous decision making for a driver-less", in Proc. of 12th IEEE Conference on Systems and System Eng. (SoSE) Conference, Waikoloa, Hawaii, 18-21 June 2017, pp. 1-6.

[9] *A.D. Ciotec, V. E. Neagoe, A. P. Bărar,* "Concurrent Self-Organizing Maps for Pedestrian Detection in Thermal Imagery", Scientific Bulletin of the Polytechnic University of Bucharest, Series C, **Vol. 75**, Iss. 4, 2013, ISSN 2286-3540.

[10] *Dollar, P., Wojek, C., Schiele, B., & Perona, P.* Pedestrian detection: An evaluation of the state of the art. IEEE transactions on pattern analysis and machine intelligence, 2011, 34(4), 743-761

[11] *V. Neagoe, M. Vâlcu, and B. Sabac*, "A Neural Approach for Detection of Road Direction in Autonomous Navigation", in: Computational Intelligence, Theory and Applications, (ed. B. Reusch), Elsevier, Berlin-New York, 1999, pp. 324-333.

[12] *V.E. Neagoe, C.T. Tudoran,* "A Neural Machine Vision Model for Road Detection in Autonomous Navigation", Scientific Bulletin of the Politehnica University of Bucharest, Series C - Electrical Engineering, No 2, 2011, pp. 167-178.

[13] *J. Yosinski, J. Clune, Y. Bengio, H. Lipson*, "How transferable are features in deep neural networks?", Advances in Neural Information Processing Systems 27, Dec. 2014, pp 3320-3328.

[14] *A. Krizhevsky, I. Sutskever, G. E. Hinton.* Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp 1097–1105

[15] *G. Jocher, A. Stoken, J. Borovec, NanoCode012, C. STAN, L. Changyu, ... L. Yu 于力军.* ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration (Version v4.0). (2021, January 5). Zenodo. http://doi.org/10.5281/zenodo.4418161

[16] *P.Tumas, A. Serackis,* "Automated Image Annotation based on YOLOv3", in Proc. 6th IEEE Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), 8-10 Nov. 2018, pp. 1-3

[17] *F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, ... & T. Darrell "*BDD100k: A diverse driving dataset for heterogeneous multitask learning", In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 2636-2645), 2020

[18] *G. Jocher, A. Stoken, J. Borovec, NanoCode012, A. Chaurasia, TaoXie, ... F. Ingham.* ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations. Zenodo. http://doi.org/10.5281/zenodo.4679653, 2021

[19] *P. Bhargava, "*On Generalizing Detection Models for Unconstrained Environments", In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019

[20] *D. A. Tran, P. Fischer, A. Smajic, Y. So,* Real-time Object Detection for Autonomous Driving using Deep Learning, Institute of Computer Science, Department of Computer Science and Mathematics, Goethe University Frankfurt, 15 Mar 2021

[21] JSON2YOLOv5. https://github.com/Delphi89/JSON2YOLOv5, 2021