Towards Real-Time Traffic Sign Recognition via YOLO on a Mobile GPU

N S Artamonov¹ and P Y Yakimov¹

¹Samara National Research University, Moskovskoye Shosse 34A, Samara, Russia, 443086

e-mail: mail.artamonov@gmail.com

Abstract. Classification of objects in the video stream with the help of deep learning has gained immense popularity nowadays. Considering many systems solving the classification problem, the mobility is often required. This paper proposes an implementation of the YOLO (You Only Look Once) convolutional neural network to solve the problem of classification of traffic signs on the mobile platform NVIDIA Jetson. The main feature of this platform is the availability of mobile graphics processor NVIDIA Tegra, which allows high-performance computing with low power consumption. The implemented algorithm of the YOLO CNN neural network allows solving the problem of the classification of traffic signs in a continuous video stream with decent accuracy and speed, and the NVIDIA Jetson platform provides mobility of the system.

1. Introduction

Currently, the popularity of pattern recognition systems, integrated into various mobile objects, is increasing. Such objects include various unmanned flying and ground vehicles, such as quadrocopters or self-driving cars. Integrated in cars to solve variety of tasks, such systems are aimed to assist a driver or even replace him. One of the main problems is finding suitable hardware with low power consumption, but sufficient computing power for real-time recognition. The software also plays a great role, because you need to extract the maximum performance from the few resources that a mobile system can provide.

One of the popular hardware solutions for pattern recognition in cases where low power consumption is required is the NVIDIA Jetson mobile hardware platform based on the NVIDIA Tegra processor. NVIDIA Tegra is a family of systems on the chip (SoC), combining the ARM-processor, graphics processor, media and DSP-processors, memory and peripheral controllers, while having low power consumption. The mobile platform NVIDIA Jetson TX2 is currently used by the company "Rosteletronics" in the video recording of traffic violations [1]. Figure 1 shows an image of the NVIDIA Jetson TX2 platform, which runs with the NVIDIA Pascal GPU.

Nowadays, lots of researchers use various convolutional neural networks to process images for both recognition and detection tasks. In paper [2], the authors provide the algorithm for the super-resolution task, the task for real-time face identification is solved in [3]. Solutions for real-time image recognition are divided into two general types: Region Proposal (one by one the regions of a frame are proposed and classified) and Single Shot (all objects are simultaneously recognized in the whole image). The first type includes such neural networks as R-CNN [4], Fast R-CNN [4], Faster R-CNN
The second one includes YOLO CNN [5], SSD [6]. Neural networks using recognition by region have a rather slow recognition time for qualitative detection of objects. However, for mobile platforms, Single Shot CNNs are more suitable, as they are quite faster.

Recognition of traffic signs is usually solved in two steps: localization and subsequent classification, which is more like the described above Region Proposal type. In papers [7] and [8], the authors proposed effective implementations of the image preprocessing and traffic signs localization algorithms, which performed in real-time. Using a modified Generalized Hough Transform (GHT) algorithm, the solution allowed to determine the exact coordinates of a traffic sign in the acquired image. Using a CNN for the recognition step, the authors in [9] still do not use the neural network for traffic signs localization in images. In this paper, we describe a Single Shot solution for both localizing and recognizing traffic signs in images.

![Figure 1. NVIDIA Jetson TX2.](image)

This article describes the process of installing, training, and using the YOLO CNN neural network on the NVIDIA Jetson TK1 mobile platform, as well as analyzing the inference performance. The use of GPU on the platform, as well as CUDA and OpenCV libraries [10], allows to use the entire computing power of the system.

2. Real-time object detection using CNN

2.1. Convolutional neural networks

The recognition and classification tasks can be easily solved using convolutional neural networks. A neural network is some mathematical model that consists of interconnected artificial neurons [11]. The network accepts the characteristic vector as input, and then sequentially passes them through the layers of the network. At the output, the probabilities of belonging to the given classes are obtained. Usually, a neural network operates with numeric, and not symbolic values.

![Figure 2. Scheme of an artificial neuron.](image)
Figure 2 shows the scheme of an artificial neuron [12]. As an input, it considers the parameters $x_n$, which are either initial data or output parameters of other neurons. Each parameter has a weight, $w_{1j}, w_{2j}, w_{3j}, \ldots, w_{nj}$, which is a multiplier of every parameter. Then the weighted parameters are summed using some function transfer function. The resulting value is sent to the ion, which, after calculating the result, decides whether to transmit the signal to the next neuron. The output value will act as one of the parameters in another neuron.

At present, convolutional neural networks are most effective for image processing. A two-dimensional image is applied to the input of the convolutional neural network, which is then processed by convolutional layers. The convolutional layers transform the image fragments into a feature map.

2.2. YOLO

YOLO CNN [5] is a convolutional neural network that allows to detect and classify objects in the form of bounding boxes. Such bounding box is the minimum sized rectangle, which will contain the whole found object. YOLO works on the principle of Single Shot. This means that the network architecture is arranged in such a way that in one pass of the frame, all objects are detected simultaneously.

Figure 3 shows the architecture of YOLO CNN. The YOLO input is provided with a three-channel image, which is resized to 448x448. The first conversion is to run the image through a portion of the modified GoogLeNet architecture. After this conversion, we get the feature maps with the size 14x14x1024. Then, two convolutions are applied. After the second convolution, the dimension decreases to 7x7x1024. Then, another convolution is performed. The result is twice used in a fully connected layer, changing to a dimension of 1470x1 and is transformed into a tensor of 7x7x30. The obtained tensor is subjected to a detection procedure, at the output of which a resultant detection is obtained. The tensor is a 7x7 mesh display in the image. 30 values carry information about the cell: 10 values describe two possible frames; 20 values show the relation to each of the 20 available classes. All this information is filtered, the filtered data is displayed.

Figure 3. Architecture of YOLO CNN.

3. Implementation of the work of YOLO on NVIDIA Jetson

NVIDIA Jetson is a family of mobile platforms based on NVIDIA Tegra processors. To date, there are four generations:

- NVIDIA Jetson TK1;
- NVIDIA Jetson TX1;
- NVIDIA Jetson TX2;
- NVIDIA Jetson Xavier.

Table 1 shows the main characteristics of these models.
This article presents an implementation of the YOLO CNN on the platform NVIDIA Jetson TK1, which is the cheapest model in the family. Before you start working with the platform, you need to install all necessary software on it. This procedure is performed using Jetpack JDK, the official NVIDIA utility. To implement neural networks inference and recognition of images on the platform, the versions of the CUDA 6.5, CUDNNv2, OpenCV frameworks adapted for the hardware are provided.

YOLO CNN provides the ability to compile and launch the network using CPU, GPU, CUDA, CUDNN, OpenCV, OpenMP and their combinations. During the launch of YOLO on Jetson TK1 several problems emerged:

- The platform supports an older version of CUDNN.
- The platform does not support OpenMP.
- A small amount of RAM must be compensated with a swap file (1GB).

All these problems negatively affect the performance of the neural network. However, compilation with a combination of GPU, CUDA, OpenCV is successful. Figure 4 shows the result of YOLO on the test image.

Figure 4. Objects recognition result in YOLO.

To start YOLO, you need a configuration file (yolo.cfg), a weight file for all the required classes (yolo.weights), and a media file (an image or a video), and you can also start the video stream from the webcam). [13] Command example:

```
./darknet detect cfg ./yolo.cfg yolo.weights data /dogs.png
```

The neural network has a large set of already trained classes, which can help to test all the capabilities of YOLO, as well as the performance of hardware. Also, YOLO can be trained for any class of images, if you correctly select the source data. To train YOLO, a large dataset is required. Each image should be provided with a text file with the marked regions of the trained class of objects. You also need a file with initial weights and a file with system information, which shows the paths to the images and the path, through which the recovery points will be recorded. Recovery points are the

### Table 1. Main characteristics of the platform NVIDIA Jetson.

|          | Jetson TK1                          | Jetson TX1                          | Jetson TX2                          | Xavier        |
|----------|-------------------------------------|-------------------------------------|-------------------------------------|---------------|
| **GPU**  | NVIDIA Kepler                      | NVIDIA Maxwell                      | NVIDIA Pascal                       | 512-core Volta GPU with Tensor Cores |
|          | GPU with 192 cores CUDA             | TM with 256 cores CUDA              | with 256 cores CUDA                 |               |
| **CPU**  | 4-core ARM® Cortex™-A15             | 4-core ARM® A57/2 MB L2             | HMP Dual Denver                      | 8-core ARMv8.2 64-bit CPU, 8MB L2 + |
|          | with architecture NVIDIA 4-Plus-1™ |                                    | 4-core ARM® A57/2 MB L2             | 4MB L3        |
| **Memory** | 2 GB of memory x16, 64-bit bus      | 4 GB of memory LPDDR4, 64-bit       | 8 GB of memory LPDDR4, 128-bit      | 16GB 256-bit |
|          |                                     | 25.6 GB/s                           | 59.7 GB/s                           |               |
|          |                                     |                                     |                                     |               |
files of the weights at a certain training step. Weighting files are written to permanent memory every 100 iterations, which allows you to interrupt training at any time, and then continue with the last received weight. The longer the network is trained, the better the detection quality will be. The command for training YOLO is as follows:

```
./darknet detector train cfg / voc.data cfg / yolo-voc.cfg darknet.conv.23
```

It is not necessary to use the mobile platform to train the YOLO neural network. For this purpose, a computer with a powerful GPU is more suitable. Thus, the Jetson platform will be used to directly apply the trained neural network.

It should also be noted that the performance of NVIDIA Jetson TX1, TX2 or Xavier significantly exceeds the NVIDIA Jetson TK1 used in this study. However, the large software compatibility of different generations of hardware for NVIDIA products allows to try the methods and algorithms on the available hardware for launching on newer platforms.

To train the neural network, we created a new dataset. The training procedure needs images in raster format (for example, .jpg) and marked areas for them with objects in a separate text file with the same name as the image. The format of the entries in the file is as follows:

```
class_number x_center_of_region y_center_of_region width_of_region height_of_region
```

Also, for training purposes, we used the German Traffic Signs Recognition Benchmark dataset [14]. This is a large database with German traffic signs. It contains 43 different classes of traffic signs. The number of images is more than 50000. Separately, it contains data for training and for tests. All regions are provided in the .csv format. This format is easy to convert to YOLO format. One of the main disadvantage of the dataset is the absence of data augmentation.

Data augmentation is the process of extending the initial dataset using various techniques, including images rotating, resizing, retouching, etc. In our study, we didn’t use augmentation. But it's easy to do by copying the same image by turning from different angles. Then the resulting images can be mirrored. You can change the brightness and contrast of each picture with a certain step. Such copying usually increases the amount of data several times.

4. Experimental results
To solve the problem of the classification of traffic signs, two datasets were prepared to train YOLO. The first dataset is self-made and consists of 130 images of the "Crosswalk" sign. The neural network performed 900 iterations of training, then the resulting weights were used to detect the image of the sign. It took 60 minutes to train the network using Intel Core i5-4670 and Nvidia GeForce GTX 1050ti. Figure 5 shows the result of detection. The training was successful, despite the small amount of initial data and iterations during training.

![Figure 5. The result of detecting the sign "Crosswalk".](image)
The second dataset is taken from the German Traffic Sign Recognition Benchmark [14]. The training of the neural network took more than 24 hours using Intel Core i5-4670 and Nvidia GeForce GTX 1050ti. More than 10,000 iterations were executed, which is quite a good amount for image recognition tasks. Figure 6 shows the results of recognizing the sign “priority road”.

![Figure 6. Results of detecting the sign "priority road".](image)

However, there are still cases of incorrect recognition. This is due to various factors: the larger traffic signs datasets are required, and all the traffic signs type are included in the dataset used for training. Figure 7 shows the case of the sign “circular motion” being recognized as the sign “go right”.

![Figure 7. The wrong result of detecting the sign "circular motion".](image)

Table 2 shows the performance results of YOLO implemented on NVIDIA Jetson TK1 and on a PC with a GPU GeForce GTX 950 and GeForce GTX 1050ti, and trained with German Traffic Signs Recognition Benchmark dataset and image of “stop” sign from Figure 8.

| NVIDIA Jetson TK1 | NVIDIA GeForce GTX950 | NVIDIA GeForce GTX1050ti | GeForce |
|-------------------|------------------------|---------------------------|----------|
| Time of recognition of one image in seconds | 1.71453               | 0.04220                   | 0.04159  |
| Frames per second with video stream          | 1.5-2.2               | 18-22                     | 18-22    |
Figure 8. The "stop" sign image and result of it recognition.

Obviously, the recognition speed on the NVIDIA Jetson TK1 is lower than on the desktop GPUs. This is because the platform has little RAM. When the neural network is operating, all RAM is filled, and the system connects the operation of the swap file. Also, the platform doesn’t utilize CUDNN and OpenMP frameworks.

The results of the recognition time of a single image do not directly depend on the fps in the video stream. This is because the image size for the tests does not coincide with the size of the video stream frames.

The GTSRB dataset was first introduced in 2011 and has been used for research in the TSR area for years. Still being quite useful for training new CNN models, the German dataset focuses on the German traffic signs, which constricts the potential application of the developed system. In future, we plan to use some other datasets like the Russian Traffic Sign Images Dataset [15].

5. Conclusion

This paper describes the implementation of the YOLO CNN on the mobile platform NVIDIA Jetson. Trained for traffic signs detection, the network shows good results and verifies the ability of using YOLO for real-time image processing on a mobile platform. While the CNN inference on Jetson TK1 is rather slow, it shows good quality of objects recognition. To satisfy the real-time condition, more powerful devices like Jetson TX2 or Jetson Xavier are needed. Thus, with low consumption of up to 30 watts and being quite affordable, such mobile platform combined with YOLO is a good choice for advanced driver assistance systems.

6. References

[1] 3D News. Daily Digital Digest (Access mode: https://3dnews.ru/959458) (01.11.2017)
[2] Nikonorov A V, Petrov M V, Bibikov S A, Kutikova V V, Morozov A A and Kazanskiy N L 2017 Image restoration in diffractive optical systems using deep learning and deconvolution Computer Optics 41(6) 875-887 DOI: 10.18287/2412-6179-2017-41-6-875-887
[3] Vizilter Yu V, Gorbatevich V S, Vorotnikov A V, Kostromov N A 2017 Real-time face identification via CNN and boosted hashing forest Computer Optics 41(2) 254-265 DOI: 10.18287/2412-6179-2017-41-2-254-265
[4] Ren S, He K, Girshick R, Sun J 2016 Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks p. 14
[5] Redmon J, Divvala S, Girshick R, Farhadi A 2015 You Only Look Once: Unified, Real-Time Object Detection You Look Only Once p. 10
[6] Liu W, Anguelov D, Erhan D,Szegedy C,Reed S,Fu C,Berg A 2016 SSD: Single Shot MultiBox Detector p. 15
[7] Yakimov P Y 2015 Tracking traffic signs in video sequences based on a vehicle velocity Computer Optics 39(5) 795-800 DOI: 10.18287/0134-2452-2015-39-5-795-800
[8] Fursov V A, Bibikov S A, Yakimov P Y 2013 Localization of objects contours with different scales in images using Hough transform Computer Optics 37(4) 496-502
[9] Shustanov A, Yakimov P 2017 CNN design for real-time traffic sign recognition Procedia Engineering 31(201) 718-775
[10] OpenCV Image Processing Library (Access mode: http://opencv.org) (04.11.2017)
Acknowledgments
This work was supported by the Russian Foundation for Basic Research - Project # 16-37-60106 mol_a_dk.