Car model classification using NVidia DIGITS and GoogLeNet

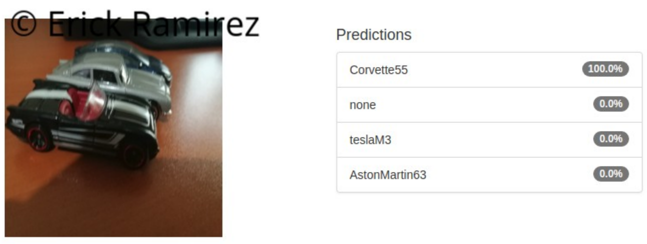
Erick Ramirez

Abstract—This paper describes the implementation of a car model classification using deep learning on the Nvidia DIGITS 6, for this specific solution the GoogLeNet architecture. Three models have been used and a forth class has been included as none which is the absence of cars.The dataset created was based on a series of videos. Upon the evaluation, the created model recognized the cars correctly.

Index Terms—Robot, IEEEtran, Udacity, deep learning, NVIDIA

1 Introduction

Back in 2012, AlexNet was the first Deep Neuronal network architecture which won the ImageNet classification challenge [1], since that moment Deep neuronal networks were back on track for computer vision. Currently there are more DNN architectures, which are more complex and accurate than AlexNet. In this work, deep learning techniques were utilized to create a classification solution for car models. The solution uses NVIDIA Deep Learning GPU Training System (NVIDIA DIGITS) and it allows to train datasets with existing pre-trained models, in this case: LeNet, AlexNet and GoogLeNet. An application for this will be in toll roads, normally rate is for vehicle type and set this rate according with the vehicle and then open and close the bar/gate, also obtain information from the type of vehicles. in Guatemala there are toll roads and there are several persons operating this gates and normally it evaluates a serializable vehicles (only one each time). For this paper, GoogLeNet has been used as model for the task of classification car models, because of this architecture won the ImageNet classification challenge in 2014 (ILSVRC 2014). [2]



```
Predictions
Corvette55      100.0%
none              0.0%
teslaM3           0.0%
AstonMartin63     0.0%
```

Fig. 1. The three toy cars in the same images, classified as corvette

2 Background / Formulation

Deep learning for computer vision is commonly used for task like classification, object detection, semantic segmentation, and so on. It has been adopted to many applications and it is because of the computer power (specifically the GPU), software and the access to amount of available data that we have. In a neuronal network there are several hyper parameters and layer combinations, which will impact in the final output. The hyper parameters used are the following:

- Epoch: is one pass of the full training data set. Normally a small number of epoch is not good enough for the task, because can cause bad predictions and a big number of epoch will mean more training time and a possible over fitting of the data. In this case the selected value is 15.
- Learning rate: this controls how much we are adjusting the weights of our network with respect the loss gradient. For this case, the initial learning rate is 0.0005.
- Solver type: This is the optimizer and it contains information about how the weights are updated for the network.

GoogLeNet uses several techniques known from for convolutional neuronal networks, such as: convolutions, pooling, adding softmax and so on. It performs the feature extraction and then the classification.

3 Data Acquisition

The suplied dataset was taken from videos on 3 toy cars, which models are the following:

- Chevrolet Corvette 1955
- Tesla Model S 2014
- Aston Martin 1963

This is the data distribution:

TABLE 1
data distribution

Car Model	Number of images
Chevrolet Corvette 1955	4049
Tesla Model S 2014	3246
Aston Martin 1963	3761
None (no car present)	1060

TABLE 2
Real name of classes

Real name	Name in the model
Chevrolet Corvette 1955	Corvette55
Tesla Model S 2014	TeslaM3
Aston Martin 1963	AstonMartin63
No car present	None

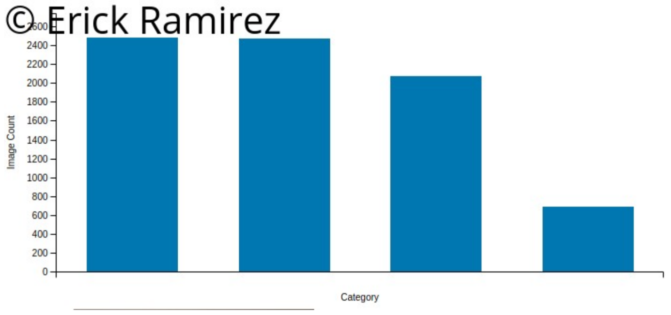


Fig. 2. Data distribution for the training

Real name for each class: The data to be classified had been labeled with a short name,the entire name of each class is the following.

The data for the validation is a subset of the data uploaded, 25% for data validation and 10% for testing. The original resolution of the image is 1920x1080 (FHD) and it was taken with a cellphone camera. I converted the videos in images and resized them in the resolution 256x256 pixels and I selected the valid images for each car. The videos provided information from different angles, distances and background of the toy cars. The data can be acceded by S3 in https://s3-us-west-2.amazonaws.com/test-erick/CarModelData.tar.gz.

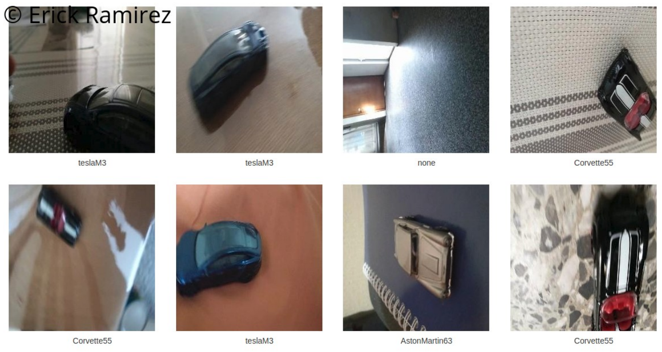
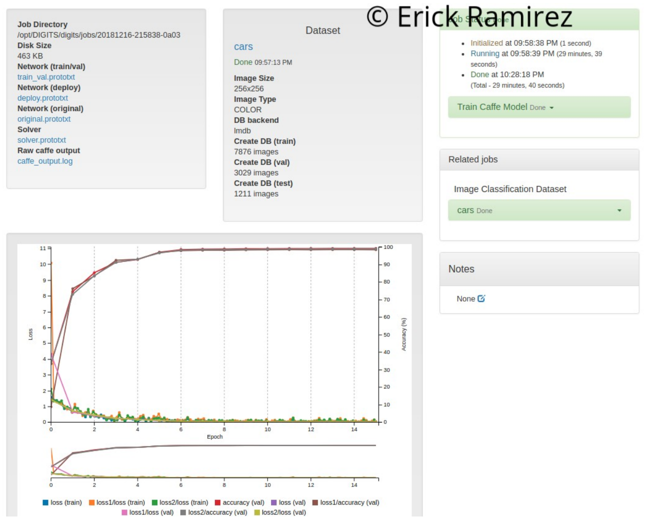


Fig. 3. Examples of data uploaded to the model, this can be downloaded in https://s3-us-west-2.amazonaws.com/test-erick/CarModelsimagenes.tar.gz.

In this case I selected a pretrained model, the other big factor is the data collected. That is why I performed a review of the images obtained from the video and deleted the ones that doesn't have the car anymore. In this case the images are png and in RGB color model, I keep it this way because I was interested in validate the color and grayscale also check the color because of the scale color.

4 Results

The final classification model was gathered from running the described image set over 10 epochs of GoogLeNet modelon Nvidia DIGITS, using stochastic gradient descent as solver type. You can check the accuracy of this model, which is over 99% of accuracy (99.24% for the training), the final value loss was around the 0.0279735% and the learning rate used is 0.0005.



Job Directory
/opt/DIGITS/digits/jobs/20181216-215838-0a03
Disk Size
463 KB
Network (train/val)
train_val.prototxt
Network (deploy)
deploy.prototxt
Network (original)
original.prototxt
Solver
solver.prototxt
Raw caffe output
caffe_output.log

Dataset
cars
Done 09:57:13 PM
Image Size
256x256
Image Type
COLOR
DB backend
lmdb
Create DB (train)
7876 images
Create DB (val)
3029 images
Create DB (test)
1211 images

- Initialized at 09:58:38 PM (1 second)
- Running at 09:58:39 PM (29 minutes, 39 seconds)
- Done at 10:28:18 PM (Total - 29 minutes, 40 seconds)

Related jobs
Image Classification Dataset
cars Done

Notes
None

Fig. 4. Training resume, it passes the required accuracy of 75%.

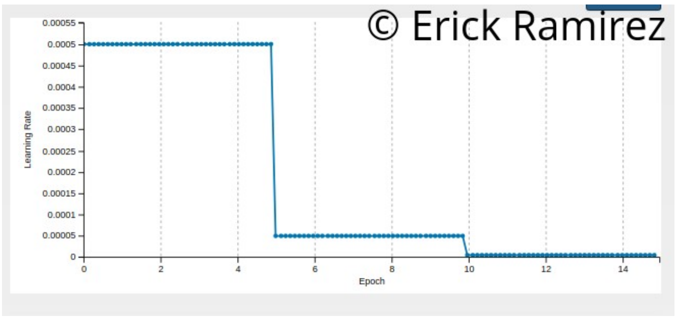


Fig. 5. Learning rate.

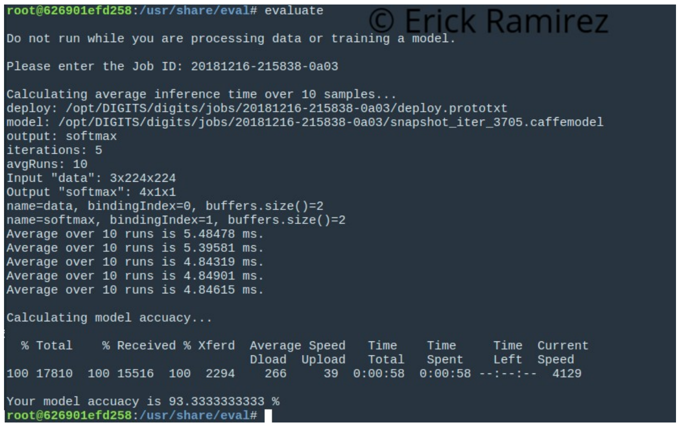
Different combinations were tested, for instance Adam as solver type and I got a 40% of accuracy, but sometimes it got less accuracy or overfitting. You can check the following figure, where it shows the prediction accurately in more this range 85% - 100%.

Test with other colors:

The prediction was right. For instance in other colors, the prediction was still right but the accuracy was lower 47% - 100%. There is a bad classification. after multiple testing on the model, it is a good for this toy car models, and even with the real ones (the testing data), however it has some overfit related to the colors of the car selected.

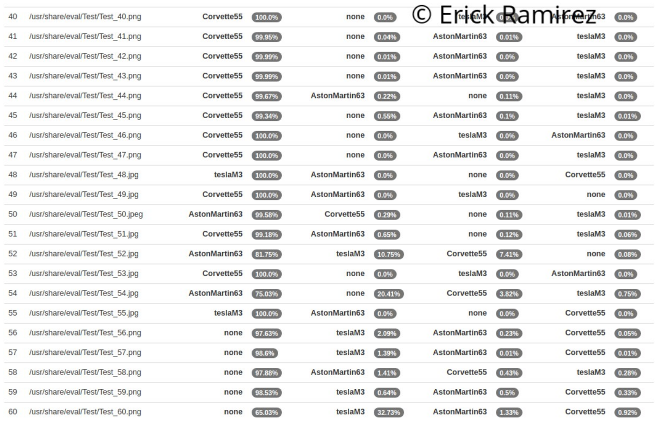
5 Discussion

Modern robots uses deep learning to interpret the sensor data, for instance the RGBD and/or RGB cameras generate images that can be processed(image detection, classification)



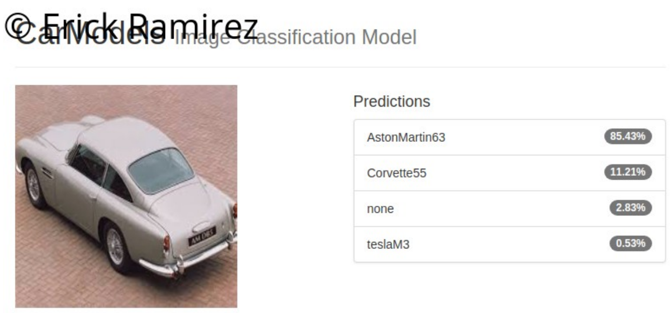
```
root@626901efd258:/usr/share/eval# evaluate

Do not run while you are processing data or training a model.

Please enter the Job ID: 20181216-215838-0a03

Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20181216-215838-0a03/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20181216-215838-0a03/snapshot_iter_3705.caffemodel
output: softmax
iterations: 5
avgRuns: 10
Input "data": 3x224x224
Output "softmax": 4x1x1
name=data, bindingIndex=0, buffers.size()=2
name=softmax, bindingIndex=1, buffers.size()=2
Average over 10 runs is 5.48478 ms.
Average over 10 runs is 5.39581 ms.
Average over 10 runs is 4.84319 ms.
Average over 10 runs is 4.84901 ms.
Average over 10 runs is 4.84615 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 17810  100 15516  100  2294    266     39  0:00:58  0:00:58 --:--:--  4129

Your model accuacy is 93.3333333333 %
root@626901efd258:/usr/share/eval# 
```

Fig. 6. Training resume, it passes the minimum required inference time (10ms) with an average of 5.083 ms. Note: the accuracy in this evaluation is and the files for test (check Test folder: https://github.com/Erickramirez/RoboND-Robotics-Inference/tree/master/Test) was update to a valid test data (never saw during the training) it got a 93.333% of accuracy. To check it the image_list.txt, Test folder and labels.csv files were modified



40	/usr/share/eval/Test/Test_40.png	Corvette55	100.0%	none	0.0%	teslaM3	[illegible]	AstonMartin63	0.0%
41	/usr/share/eval/Test/Test_41.png	Corvette55	99.95%	none	0.04%	AstonMartin63	0.01%	teslaM3	0.0%
42	/usr/share/eval/Test/Test_42.png	Corvette55	99.99%	none	0.01%	AstonMartin63	0.0%	teslaM3	0.0%
43	/usr/share/eval/Test/Test_43.png	Corvette55	99.99%	none	0.01%	AstonMartin63	0.0%	teslaM3	0.0%
44	/usr/share/eval/Test/Test_44.png	Corvette55	99.67%	AstonMartin63	0.22%	none	0.11%	teslaM3	0.0%
45	/usr/share/eval/Test/Test_45.png	Corvette55	99.34%	none	0.55%	AstonMartin63	0.1%	teslaM3	0.01%
46	/usr/share/eval/Test/Test_46.png	Corvette55	100.0%	none	0.0%	teslaM3	0.0%	AstonMartin63	0.0%
47	/usr/share/eval/Test/Test_47.png	Corvette55	100.0%	none	0.0%	AstonMartin63	0.0%	teslaM3	0.0%
48	/usr/share/eval/Test/Test_48.jpg	teslaM3	100.0%	AstonMartin63	0.0%	none	0.0%	Corvette55	0.0%
49	/usr/share/eval/Test/Test_49.jpg	Corvette55	100.0%	AstonMartin63	0.0%	teslaM3	0.0%	none	0.0%
50	/usr/share/eval/Test/Test_50.jpeg	AstonMartin63	99.58%	Corvette55	0.29%	none	0.11%	teslaM3	0.01%
51	/usr/share/eval/Test/Test_51.jpg	Corvette55	99.18%	AstonMartin63	0.65%	none	0.12%	teslaM3	0.06%
52	/usr/share/eval/Test/Test_52.jpg	AstonMartin63	81.75%	teslaM3	10.75%	Corvette55	7.41%	none	0.08%
53	/usr/share/eval/Test/Test_53.jpg	Corvette55	100.0%	none	0.0%	teslaM3	0.0%	AstonMartin63	0.0%
54	/usr/share/eval/Test/Test_54.jpg	AstonMartin63	75.03%	none	20.41%	Corvette55	3.82%	teslaM3	0.75%
55	/usr/share/eval/Test/Test_55.jpg	teslaM3	100.0%	AstonMartin63	0.0%	none	0.0%	Corvette55	0.0%
56	/usr/share/eval/Test/Test_56.png	none	97.63%	teslaM3	2.09%	AstonMartin63	0.23%	Corvette55	0.05%
57	/usr/share/eval/Test/Test_57.png	none	98.6%	teslaM3	1.39%	AstonMartin63	0.01%	Corvette55	0.01%
58	/usr/share/eval/Test/Test_58.png	none	97.88%	AstonMartin63	1.41%	Corvette55	0.43%	teslaM3	0.28%
59	/usr/share/eval/Test/Test_59.png	none	98.53%	teslaM3	0.64%	AstonMartin63	0.5%	Corvette55	0.33%
60	/usr/share/eval/Test/Test_60.png	none	65.03%	teslaM3	32.73%	AstonMartin63	1.33%	Corvette55	0.92%

Fig. 7. Test data resume.

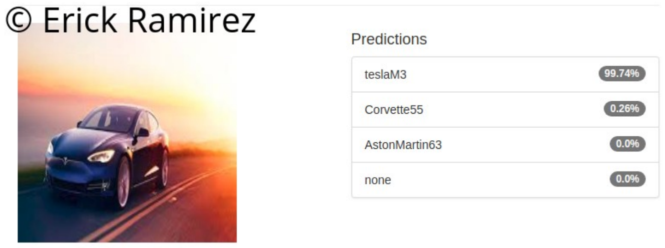


CarModels Image Classification Model

Predictions

AstonMartin63	85.43%
Corvette55	11.21%
none	2.83%
teslaM3	0.53%

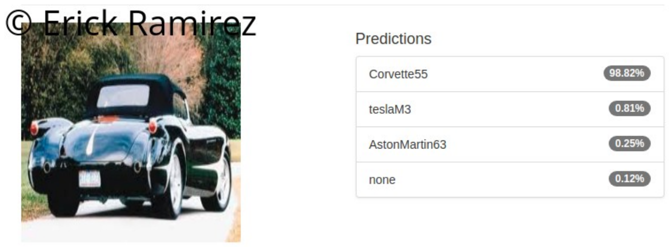
Fig. 8. Aston Martin



Predictions

teslaM3	99.74%
Corvette55	0.26%
AstonMartin63	0.0%
none	0.0%

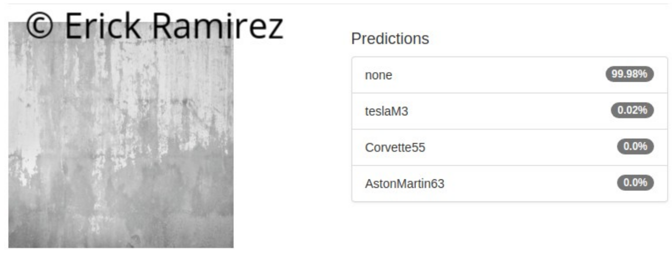
Fig. 9. Tesla



Predictions

Corvette55	98.82%
teslaM3	0.81%
AstonMartin63	0.25%
none	0.12%

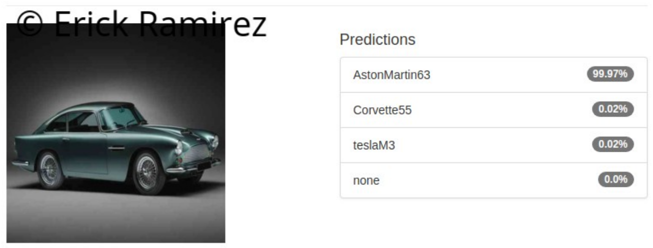
Fig. 10. Chevrolet Corvette 1955



Predictions

none	99.98%
teslaM3	0.02%
Corvette55	0.0%
AstonMartin63	0.0%

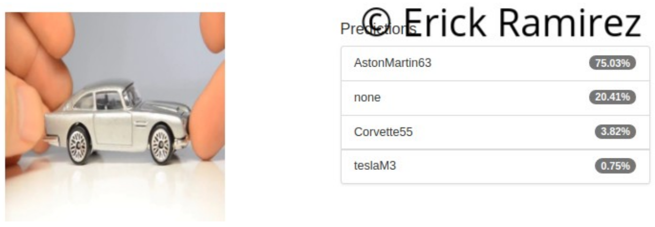
Fig. 11. None or not car present



Predictions

AstonMartin63	99.97%
Corvette55	0.02%
teslaM3	0.02%
none	0.0%

Fig. 12. Aston Martin in a different color



Predictions

AstonMartin63	75.03%
none	20.41%
Corvette55	3.82%
teslaM3	0.75%

Fig. 13. Aston Martin toy between fingers

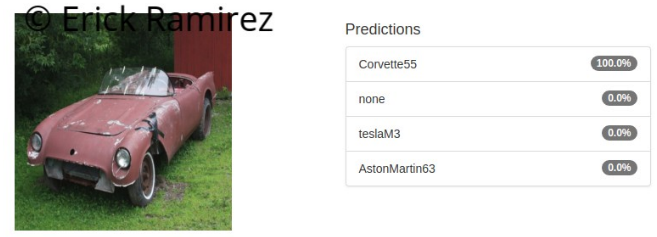


Fig. 14. Corvette in a different color

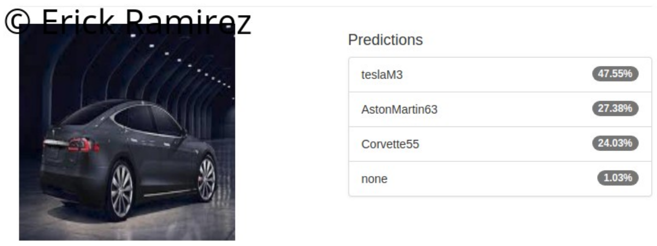


Fig. 15. Tesla on a different color

and then perform an action. The accuracy decrease with other color cars, then a more color cars can be useful. Besides that there are a good number of data uploaded, there are still positions or images that could not be correctly classified, that event the convolutional neuronal network cannot handle, different light, location or zoom, in this case it would be useful to use augmentation techniques to create more data to train. For a embedded systems, the time to respond and classify the images with GoogLeNet was fine, it is a critical aspect to evaluate when a real world problem is addressed. This model works in 201 fps which is less than 10ms on each frame (4.98 ms) [3]. There was a bad classification, it is because of the color, in some cases the images from the video didn't have the right quality of image and they didn't extract the shape of the car, in this case I think that the car was classified as other model because of the color.

6 Conclusion / Future work

The created model proves that toy car models recognition for a robotics system is possible. As previously was mentioned in the introduction in toll roads. For the four-classes, the model is working with the required accuracy which is

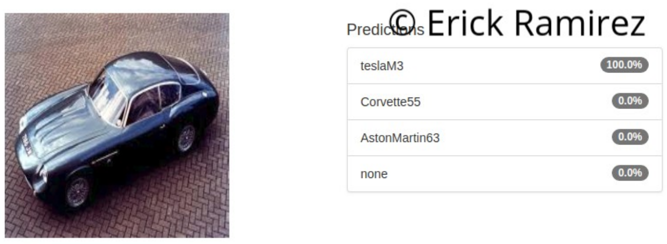


Fig. 16. Bad classification, it was classified as a Tesla, but it really is an Aston Martin

75% and it is running faster than 10ms. This project has an overfit with the car colors and the based of this were only toy cars. A future work would be the implementation of this in JetsonTx2 and as an actuator open a gate or bar in order that a car can pass. There are some actions that could be useful to work on: Generate more training data and more variety from the real world. use augmentation techniques to generate a more robust training data in order to avoid overfit. Use other DNN architectures like ResNet.

References

[1] N. Processing, "Imagenet classification with deep convolutional neural networks, howpublished = "https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks/."
[2] *Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)*. Imagenet, 2014.
[3] NVIDIA, "Nvidia jetson tx2 delivers twice the intelligence to the edge, howpublished = "https://devblogs.nvidia.com/jetson-tx2-delivers-twice-intelligence-edge/."